VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV VÝKONOVÉ ELEKTROTECHNIKY A ELEKTRONIKY
DEPARTMENT OF POWER ELECTRICAL AND ELECTRONIC ENGINEERING

SOFTWARE PRO LABORATORNÍ STAND S MĚNIČI A MOTORY
SOFTWARE FOR LABORATORY STAND WITH INVERTERS AND MOTORS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Martin Smolák

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jan Knobloch

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Silnoproudá elektrotechnika a elektroenergetika**
Ústav výkonové elektrotechniky a elektroniky

Student: Martin Smolák ***ID:*** 173743
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Software pro laboratorní stand s měniči a motory

POKYNY PRO VYPRACOVÁNÍ:

1. Dle definované struktury realizujte řídicí algoritmus pohonu.
2. Vytvořte prostředí pro realizaci laboratorních úloh.
3. Zpracujte návod pro laboratorní cvičení.

DOPORUČENÁ LITERATURA:

[1] MC56F827xx Reference Manual. NXP [online]. US: Freescale, 2013 [cit. 2016-10-24].

[2] DSC56800EX Quick Start User Guide. NXP [online]. US: Freescale, 2015 [cit. 2016-10-24].

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 31.5.2017

Vedoucí práce: Ing. Jan Knobloch
Konzultant:

doc. Ing. Petr Toman, Ph.D.
předseda oborové rady

Abstrakt

Tato bakalářská práce se zabývá návrhem řídícího firmwaru pro dva mikroprocesory a softwaru vizualizace laboratorních úloh na počítači. První z mikroprocesorů řídí stejnosměrný motor s permanentními magnety a druhý indukční motor. V práci je nejprve popsán hardware standu, princip řízení střídavých motorů, Clarkové a Parkova transformace, popis firmwaru pro stejnosměrný a indukční motor. Dále popis vizualizačního prostředí, který slouží studentům k ovládání standu a výpočet regulátoru proudu pomocí metody optimálního modulu.

Klíčová slova

indukční motor; stejnosměrný motor s permanentními magnety; laboratorní stand; mikroprocesorové řízení; digitální signálový procesor; FreeMASTER; Freescale MC56F82748; modulace prostorového vektoru; MSCAN

Abstract

This bachelor‘s thesis focuses on control firmware for two microprocessors and visualization software of laboratory tasks on the computer. The first of the microprocessor controls a DC motor with permanent magnets and the second one an induction motor. The thesis first describes the hardware of the stand, the principle of AC motors control, Clark‘s and Park‘s transformation, description of firmware for DC and induction motor. Further there is described the visualization environment which serves students for controlling the stand and for calculation of the current regulator using the method of optimal module.

Keywords

induction motor; direct current motor with permanent magnets; laboratory stand; microprocessor control; digital signal processor; FreeMASTER; Freescale MC56F82748; space vector modulation, MSCAN

Bibliografická citace:

SMOLÁK, M. *Software pro laboratorní stand s měniči a motory.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 52 s. Vedoucí bakalářské práce Ing. Jan Knobloch.

Prohlášení

„Prohlašuji, že svou závěrečnou práci na téma Software pro laboratorní stand s měniči a motory jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne **31. května 2017** ……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Janu Knoblochovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne **31. května 2017** ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Cílem této práce je navrhnout řídící systém výukového laboratorního standu, který se bude používat pro výuku v laboratořích elektrických pohonů. Laboratorní stand se skládá ze dvou pohonů, které jsou spojeny pružnou spojkou, výkonové části a řídící části.

V první části práce je stručně vysvětlené hardwarové zapojení standu. V této části lze najít parametry indukčního a stejnosměrného motoru s permanentními magnety, uspořádání jednotlivých komponentů na laboratorním standu a blokové schéma standu.

V druhé kapitole je popsána úprava hardwaru a softwaru laboratorního standu. V této kapitole je popsána komunikace sběrnice CAN, schéma zapojení, návrh desky plošných spojů, popis nastavení a inicializace periférie.

Třetí kapitola se zabývá nejčastěji používanými způsoby řízení střídavých pohonů – skalární a vektorové řízení. V této práci je prozatím použito pouze skalární řízení indukčního motoru. Vektorové řízení je v práci rovněž zmíněno. Dále jsou zde popsány algoritmy v řízení střídavých strojů, Clarkové a Parkova transformace a modulaci prostorového vektoru napětí.

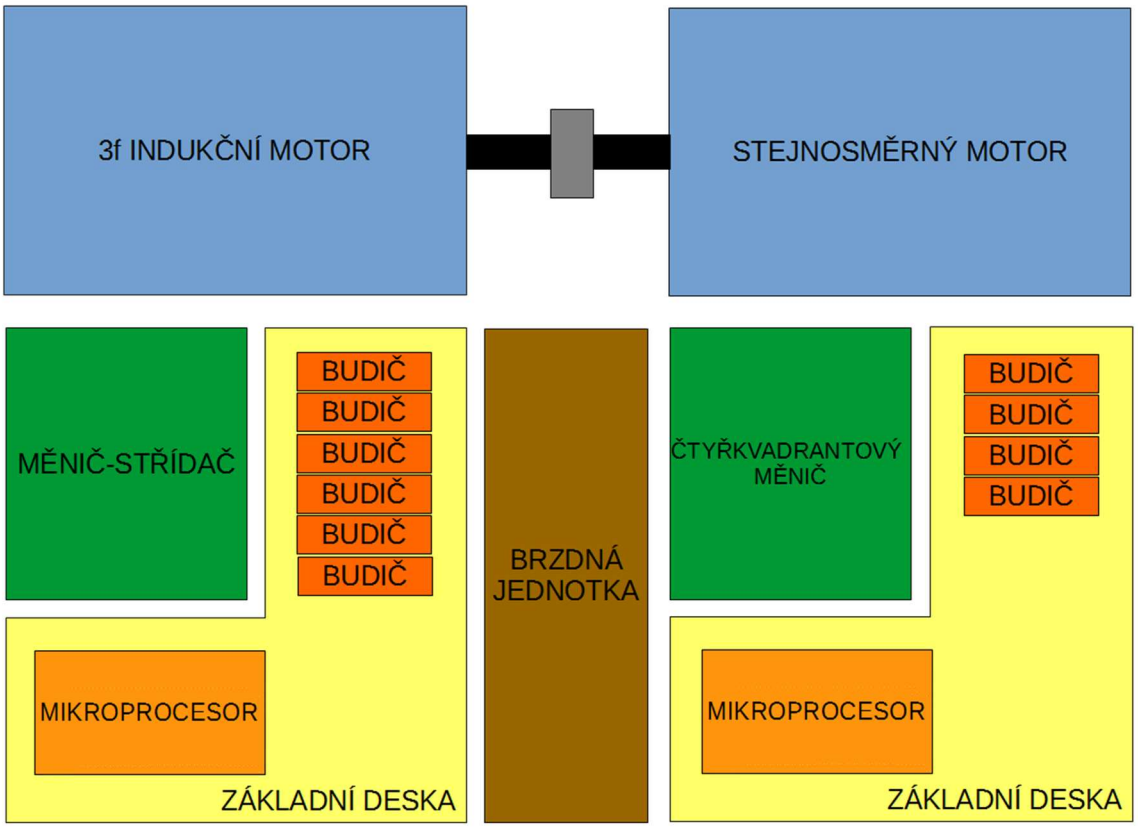
Čtvrtá část práce je rozdělena na dvě podkapitoly. První podkapitola se zabývá popisem firmwaru pro měnič stejnosměrného motoru, kde jsou popsány jednotlivé části. Druhá podkapitola se zabývá popisem firmware měniče, který je určen pro indukční motor.

V páté kapitole je popsáno vizualizační prostředí, ve kterém budou studenti měřit laboratorní úlohy. V této práci je zpracované prostředí (první laboratorní úlohy), ve které se studenti naučí navrhovat PI regulátor proudu stejnosměrného motoru.

2 POPIS HARDWARE LABORATORNÍHO STANDU

2.1 Uspořádání jednotlivých komponentů na laboratorním standu

Laboratorní stand obsahuje dva motory, které jsou spojeny pružnou spojkou. První z motorů je třífázový indukční motor v patkovém provedení a druhý je stejnosměrný motor s permanentními magnety v přírubovém provedení. Výběr a dimenzování motoru je popsáno v jiné práci [1]. Pod krytem ventilátoru indukčního motoru je umístěn resolver. Dále laboratorní stand obsahuje totožné základní desky, měniče a mikroprocesorové jednotky. Uprostřed standu je umístěna brzdná jednotka [2; 3].



Obrázek 1 Umístění jednotlivých součástí laboratorního standu – upraveno z [2]

2.2 Parametry třífázového indukčního motoru

Tabulka 1 Parametry indukčního motoru - upraveno z [1].

	Veličina	Hodnota	Jednotka
Počet pólů	$2p$	4	[-]
Jmenovitý výkon	P_n	180	W
Jmenovité napětí	U_n	24	V
Jmenovité otáčky	n_n	1350	min^{-1}
Jmenovitý proud	I_n	9,37	A
Jmenovitý moment	N_n	1,3	Nm
Účinnost	η	60	%
Účiník	$cos\phi$	0,77	[-]
Poměrný záběrný moment	M_z/M_n	1,9	[-]
Poměrný záběrný proud	I_k/I_n	3	[-]
Poměrný moment zvratu	M_{max}/M_n	2	[-]

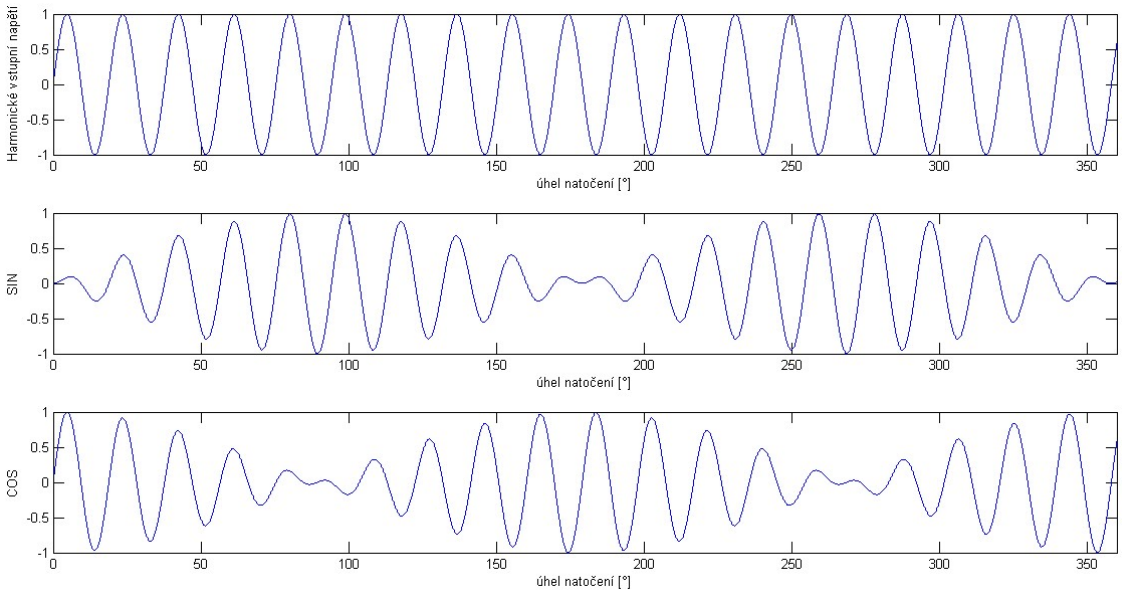
2.3 Parametry stejnosměrného motoru

Tabulka 2 Parametry stejnosměrného motoru – upraveno z [1].

	Veličina	Hodnota	Jednotka
Jmenovité napětí	U_n	24	V
Jmenovitý výkon	P_n	413,98	W
Jmenovitý moment	N_n	1,33	Nm
Jmenovitý proud	I_n	23,04	A
Jmenovité otáčky	n_n	2966	min^{-1}
Otáčky na prázdno	n_0	3449	min^{-1}
Účinnost	η	74,87	%

2.4 Snímání otáček motorů

Pro snímání polohy, popřípadě otáček je použit resolver typu ER5KC242, který je umístěný pod víkem ventilátoru indukčního motoru. Signály z resolveru jsou zpracovány R/D převodníkem AD2S1200, který je umístěný na základní desce. Tento převodník má celkem dva výstupy. Jeden z výstupů je emulovaný inkrementovaný enkodér. Druhý výstup je přes SPI rozhraní.



Obrázek 2 Vstupní a výstupní signály resolveru – překresleno z [4]

Z převodníku R/D jde do rotoru resolveru harmonický signál – viz harmonické vstupní napětí na předchozím obrázku. Indukované napětí na statoru má stejnou frekvenci jako harmonické napětí přiváděné na rotor, ale je amplitudově modulované podle úhlu natočení rotoru. Stator resolveru se skládá ze dvou cívek, které jsou vzájemně pootočeny o 90°, tudíž výstupní signál z první cívky má složku sinus a druhý výstupní signál má složku kosinus [2; 3; 4].

2.5 Chlazení indukčního motoru

Chlazení indukčního motoru muselo být provedeno elektrický ventilátorem, jelikož na místě původního ventilátoru je umístěn resolver [2].

2.6 Výkonový měnič

Laboratorní stand obsahuje celkem dvě desky s výkonovým měničem. První měnič pro indukční motor je zapojený jako třífázový střídač. Druhý měnič pro stejnosměrný motor je zapojen jako čtyř-kvadrantový pulsní měnič. Zapojení měničů umožňuje rekuperaci energie do stejnosměrného meziobvodu. Oba měniče jsou konstrukčně totožné, ale u měniče pro stejnosměrný motor jsou využity pouze dvě fáze – je vytvořen H–můstek. Deska výkonového měniče obsahuje výkonový tranzistorový modul SK115 MD 10, kondenzátory filtru, ochranu proti přepólování, dvě čidla pro měření proudu LEM LTS 25NP, které jsou umístěny ve větvích A a C (u stejnosměrného motoru senzor ve větvi C není osazen) [2].

2.7 Deska mikroprocesoru

Procesorová deska je osazena mikroprocesorem DSP MC56F82748 od firmy Freescale Semiconductor. Tento mikrokontroler je 32-bitový digitální signálový

mikrokontroler a obsahuje všechny obvody potřebné pro řízení motorů – obsahuje analogové vstupy, digitální vstupy/výstupy, čtyři PWM kanály, několik rozhraní pro komunikaci, atd. Na procesorové desce je umístěn převodník SCI/USB CP2102-GM pro sériovou komunikaci s počítačem. Procesorová deska má rozdělené napájení pro analogovou a digitální část. Procesor je napájený stabilizovaným napětím 3,3V. Deska dále obsahuje analogovou napěťovou referenci tvořenou obvodem REF2930, 8 led diod, které jsou připojené přes inventor ULN2803 na výstupy mikrokontroleru (PWM výstupy), čtyři tlačítka, z toho je jedno použito na reset mikrokontroleru a kvazi-integrační filtry prvního řádu pro vyfiltrování rušení analogového signálu [2; 3].

2.8 Brzdná jednotka

Brzdná jednotka je společná pro oba pohony, obsahuje napěťový meziobvod, který je pro napájení celého laboratorního standu, brzdný tranzistor a stabilizační obvod pro ventilátor indukčního motoru.

Napájení laboratorního standu je provedeno přes brzdnou jednotku, od které se rozvádí napětí k jednotlivým deskám. Aby při připojení k síti nevznikl velký proudový náraz, tak je deska osazena zpožďovacím členem, který opožděně připojí meziobvod k napájecímu zdroji. Před sepnutím zpožďovacího obvodu se meziobvod nabíjí přes rezistor. Stabilizační obvod stabilizuje napětí ventilátoru na 6V nebo 12V v závislosti na přepnutí přepínače. Obvod brzdného tranzistoru pracuje nezávisle na ostatních prvcích standu. V případě, že se v meziobvodu objeví přepětí – vyšší napětí než 35V – v důsledku rekuperace kteréhokoliv pohonu, tak brzdný tranzistor změní přebytečnou energii v teplo [2].

2.9 Budící obvod výkonového modulu

Budící obvody jsou umístěny na základní desce. Základní deska obsahuje celkem 6 budících obvodů pro indukční motor a 4 budící obvody pro stejnosměrný motor.

Budící obvod zajišťuje galvanické oddělení napájecího napětí a řídícího signálu. Galvanické oddělení napájení je provedeno pomoci impulsního transformátoru v DC/DC měniči. Pro galvanické oddělení řídícího signálu je budič osazen obvodem ACPL-333J, ve kterém jsou optočleny, které oddělují řídící signál a poruchový signál – saturační ochrana [2; 5].

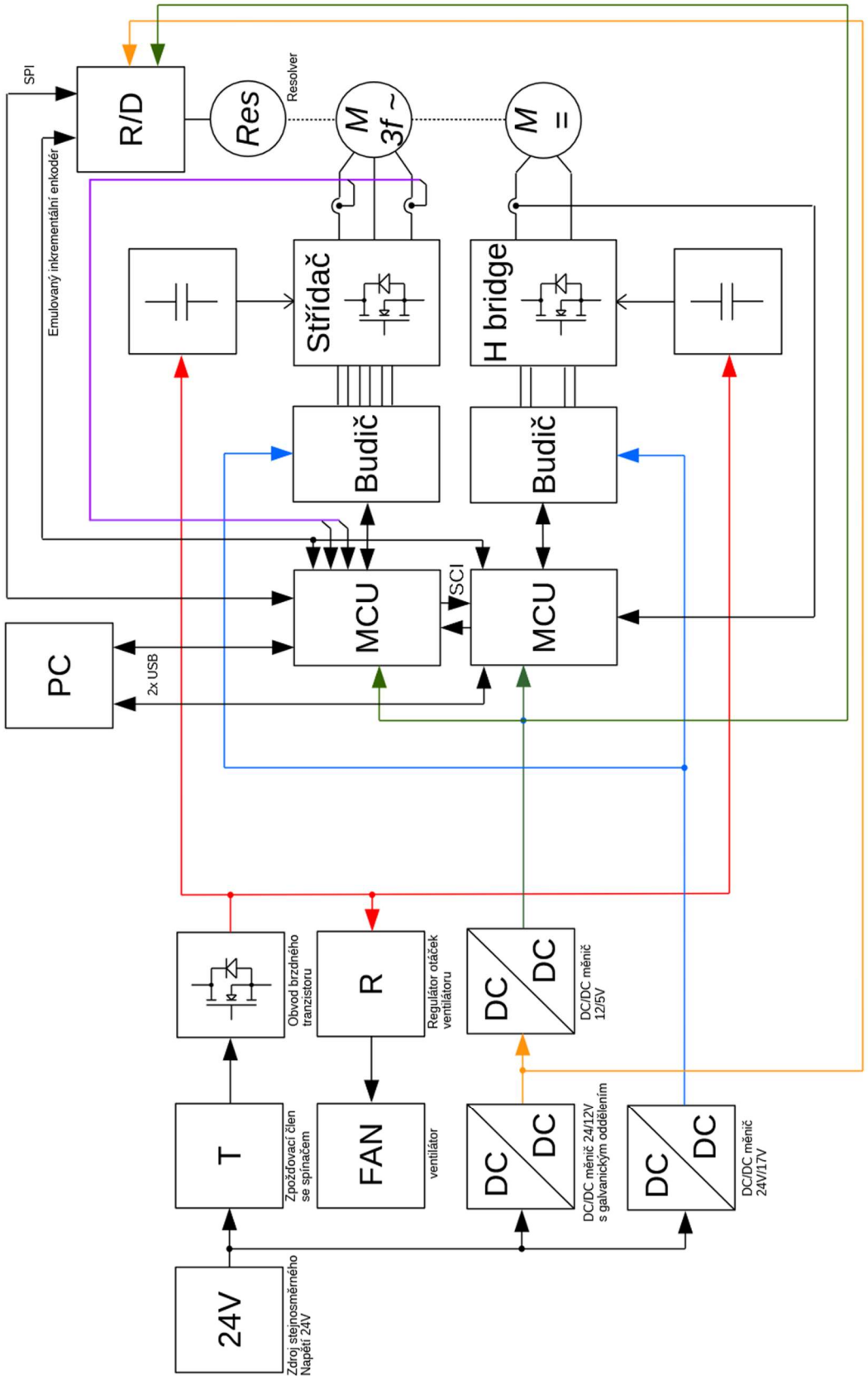
2.10 Základní deska

Základní deska je hlavní částí pro ostatní desky obsažené v laboratorním standu, jelikož zajišťuje napájení řídících obvodů. Jak bylo již zmíněno, tak základní deska je osazena R/D převodníkem AD2S1200. Na desce jsou ještě obvody pro přizpůsobení analogových signálů, vypínač pro blokování PWM signálu, který slouží pro

odlaďování programu, doplňkových logických obvodů a připojovacích konektorů [2].

2.11 Blokové schéma laboratorního standu

Blokové schéma (Obrázek 3) je navrženo podle diplomové práce ONDREJČEK [2]. Kompletní schéma je možné nalézt v uvedené diplomové práci a schéma modifikované procesorové desky je zveřejněné v bakalářské práci DRÁB [3].



Obrázek 3 Blokové schéma laboratorního standu.

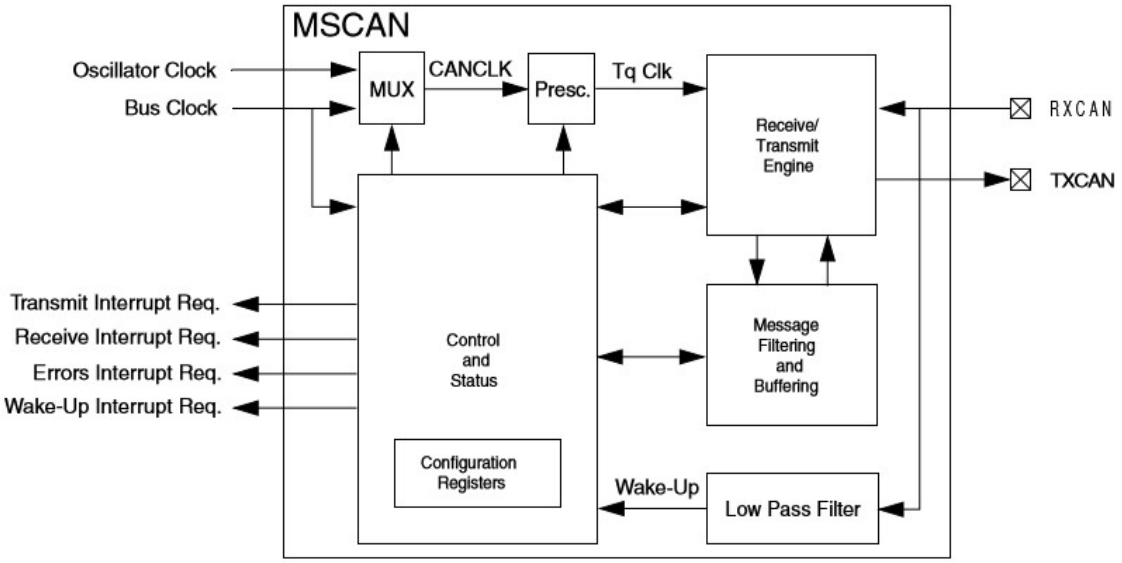
3 ÚPRAVA HARDWARE A SOFTWARE KOMUNIKACE MEZI MIKROKONTROLERY

Během testování firmware pro oba měniče bylo zjištěno, že komunikace mezi měniči je nedostačující. Původní rozhraní pro komunikaci zajišťovala periférie sériové komunikace (SCI). Jelikož přerušení po přijmu dat mělo nejnižší prioritu, tak se obsluha přerušení s vyšší prioritou vykonala během přerušení při přijmu dat => některé bajty se nepřijaly a docházelo k tomu, že data nebyla validní. Při ověřování bylo zjištěno, že procentuální validita dat byla pod 0,1 %, což je nedostačující. Z tohoto důvodu byla změněna periférie pro komunikaci z SCI na periferii MSCAN.

V periférii MSCAN je implementován protokol CAN 2.0A/B, který byl definovaný firmou Bosch v září 1991 [6]. Velká výhoda této periférie je ta, že propojení zařízení se provádí dvouvodičovou sběrnicí. Pro propojení mikrokontroleru se sběrnicí musel být navrhnut převodník s obvodem 82C251, který umožnuje připojení více zařízení na sběrnici s napětím sběrnice až ± 36V. Přenosová rychlost sběrnice je až 1Mbit/s. V našem případě je zvolena přenosová rychlost 500kbit/s [7]. Tato sběrnice je odolná proti rušení, jelikož síťový protokol detekuje a opravuje přenosové chyby, které jsou způsobeny od okolních elektromagnetických polí.

Data se na sběrnici vysílají v rámcích, který může obsahovat až osm bajtů dat. Každý rámec je označen identifikátorem *ID*, který nám říká, o jakou zprávu se jedná a určuje nám i prioritu - vyšší priorita zprávy má nižší hodnotu identifikátoru *ID* – rozlišujeme dva identifikátory. První typ identifikátoru je standartní identifikátor - 11 bitový a druhý je rozšířený identifikátor – 29bitový. My využíváme standartní identifikátor.

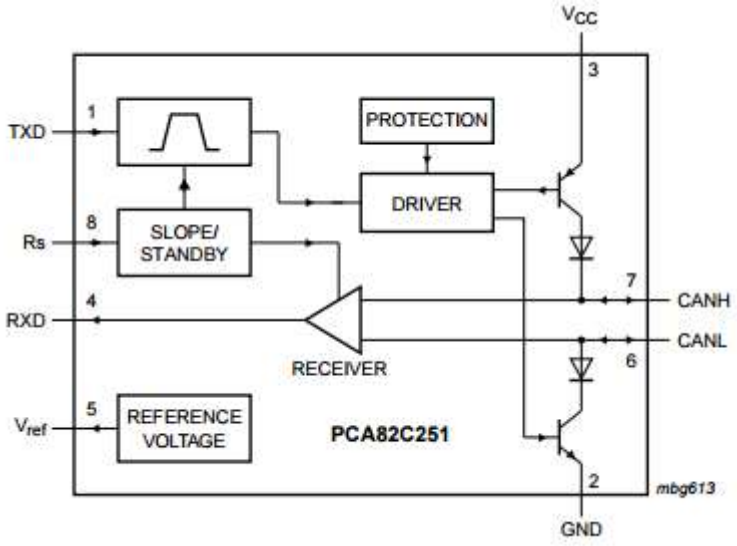
Periférie uloží do přijímacího buffru každý datový rámec. Každé zařízení může přijmout jen některé datové rámce. Příjem a vyvolání obsluhy přerušení nastává pouze u datových rámců, u kterých odpovídá *ID* nastavené přijímací masce – filtrování datových rámců [6]. Na obrázku (Obrázek 4) je zobrazen blokový diagram periférie MSCAN.



Obrázek 4 Blokové schéma periférie MSCAN - převzato z [6].

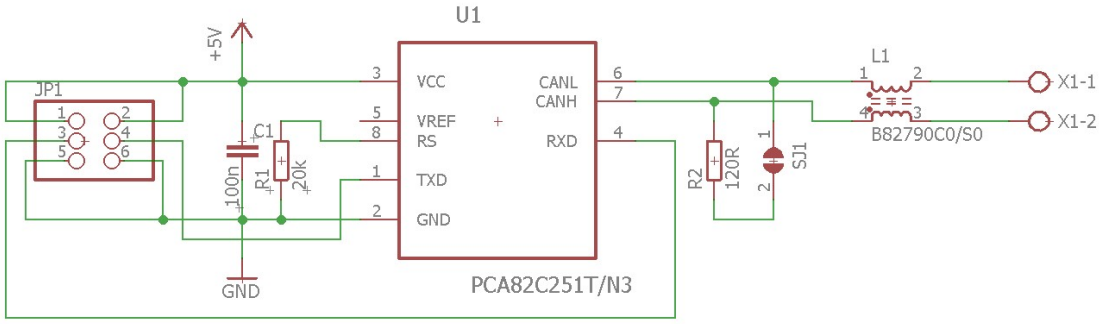
3.1 Schéma zapojení převodníku

Schéma zapojení desky CAN převodníku (Obrázek 6) obsahuje všechny nutné součástky pro správnou funkci a také součástky doplňkové, kvůli hardwarovému odrušení okolního elektromagnetického rušení. Pro převodník byl zvolen budící obvod 82C251 od firmy NXP (Obrázek 5). Základní zapojení budiče bylo doplněno o odrušovací tlumivku B82790-S513-N-3090 od firmy EPCOS pro potlačení symetrického rušení, které se může na sběrnici naindukovat. Tato tlumivka je určena přímo pro účely odrušení parazitního signálu na sběrnici CAN.



Obrázek 5 Vnitřní zapojení obvodu 82C251 - převzato z [7].

Pro universálnost převodníku bylo doplněno schéma zapojení o jumper, kterým se dá vyřadit rezistor zakončující sběrnici.

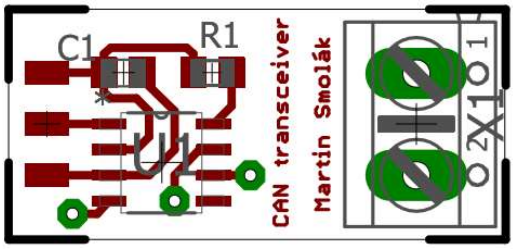


Obrázek 6 Schéma zapojení převodníku - upraveno z [7].

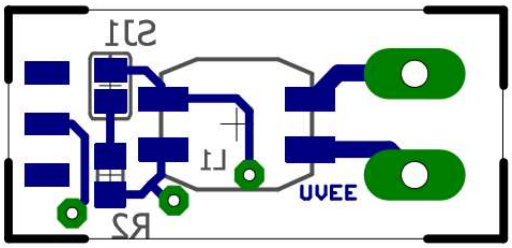
3.2 Deska plošných spojů

Desky plošných spojů jsou navrhnuty, tak aby byl rozměr desky co nejmenší (11x24mm), a aby mohla být deska zasunuta do konektoru MLW-6. Z tohoto důvodu bylo zvoleno oboustranné provedení desky plošných spojů a SMD provedení použitých součástek.

Na přiloženém CD jsou všechny soubory projektu v softwaru *EAGLE* s návrhem desky plošných spojů.



Obrázek 7 Deska převodníku – TOP – přiblíženo.



Obrázek 8 Deska převodníku - BOTTOM – přiblíženo.

Obrázek 9 Osazená deska převodníku zasunutá v konektoru MLW-6.

3.3 Softwarové nastavení periférie

Během psaní kódu pro obsluhu periférie MSCAN byl zjištěn problém s nastavením masek přijímaných zpráv. Jelikož dodávaný demo-program výrobcem mikrokontroleru je nefunkční, tak nastavení masek bylo určeno experimentálně.

Inicializace a nastavení periférie probíhá následovně. Nejprve se periférie zinicializuje, následně se nastaví mód masky. Pro naše účely nám stačí přijímání dvou zpráv – 2x (2x32 bitů). První zpráva má vyšší prioritu a druhá má nižší prioritu. Byl zvolen mód 2x32 bitových masek a standartních identifikátorů *ID*. Následně se nastaví jednotlivé masky a adresy zpráv, které mají být přijaty.

```
ioctl(MSCAN, MSCAN_INIT, NULL);
ioctl(MSCAN, MSCAN_SET_ACC_MODE, MSCAN_ACC_MODE_2X32);

ioctl(MSCAN, MSCAN_SET_ACC_MASKR_32_0, 0x0003FFFF);
ioctl(MSCAN, MSCAN_SET_ACC_IDR_32_0,
MSCAN_Id2Idr(recData.msgHiPriority.msgHi.msgCode));

ioctl(MSCAN, MSCAN_SET_ACC_MASKR_32_1, 0x0003FFFF);
ioctl(MSCAN, MSCAN_SET_ACC_IDR_32_1,
MSCAN_Id2Idr(recData.msgLoPriority.msgLo.msgCode));
```

Nyní se povolí přerušení od periférie. Pro povolení přerušení od periférie se musí vypnout *soft-reset* mode.

```
ioctl(MSCAN, MSCAN_SOFT_RESET, MSCAN_OFF);

ioctl(MSCAN, MSCAN_ERINT_SET_RSTATE_MODE, MSCAN_STAT_ALL);
ioctl(MSCAN, MSCAN_ERINT_SET_TSTATE_MODE, MSCAN_STAT_ALL);
ioctl(MSCAN, MSCAN_ERINT_ENABLE, MSCAN_RXFULL | MSCAN_STATCHNG |
MSCAN_OVERRUN | MSCAN_WAKEUP);
```

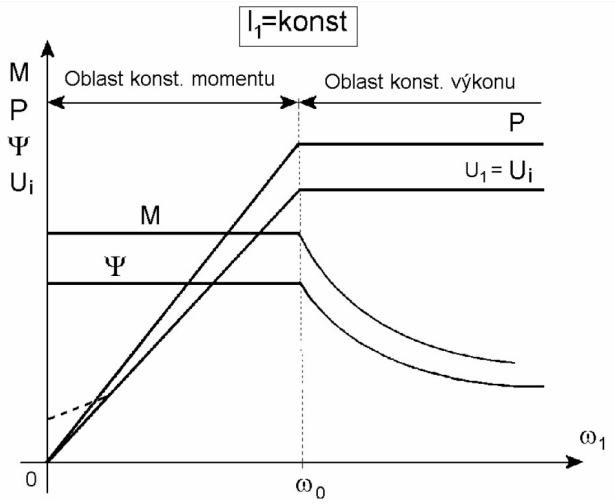
4 PRINCIP ŘÍZENÍ INDUKČNÍCH MOTORŮ

Pro napájení indukčních motoru z frekvenčních měničů jsou využívány dva základní typy metod řízení – skalární a vektorové řízení. [3].

4.1 Skalární řízení

Skalární řízení indukčních motorů můžeme použít v případě, že známe jeho statické charakteristiky. Tento typ řízení využíváme u levných pohonů, kde nejsou dány požadavky na dynamiku pohonu – nerespektují se elektromagnetické děje v indukčním stroji. Při tomto způsobu řízení využíváme nastavení frekvence a amplitudy napájecího statorového napětí – *U/f* řízení indukčního motoru.

Z regulačního algoritmu získáme amplitudu prostorového vektoru napětí a jeho natočení v daném okamžiku. Tento vektor se otáčí rychlostí ω_s. Polohu vektoru získáme integrací rychlosti ω_s. Z metody *U/f* řízení indukčního motoru vyplývá, že v pracovní oblasti momentové charakteristiky má stroj konstantní sycení dané poměrem *U/f*–v této pracovní oblasti má motor konstantní moment. Při překročení maximální jmenovité frekvence motoru se nemůže zvyšovat amplituda svorkového napětí, jelikož vyšší efektivní hodnota napětí není k dispozici. Při dalším zvyšováním frekvence dochází k odbuzování motoru – pracuje ve stavu s konstantním výkonem [8; 3; 9].



Obrázek 10 Závislost charakteristických veličin na úhlové rychlosti – převzato z [9].

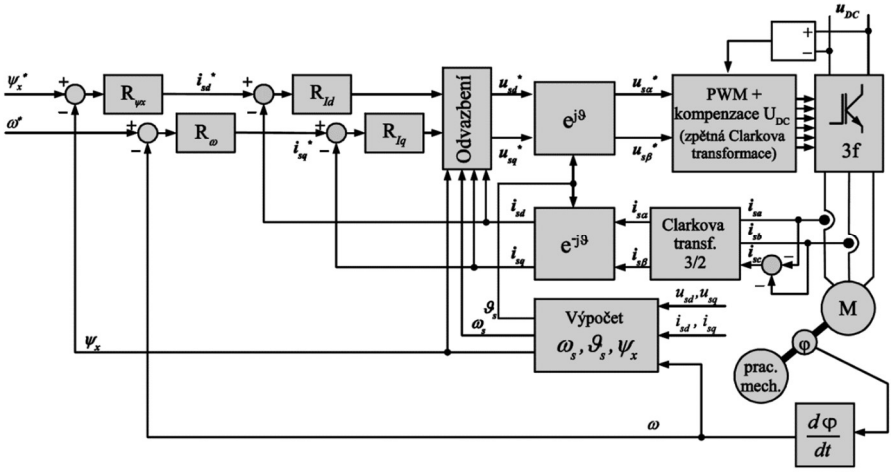
4.2 Vektorové řízení

U vektorové řízení se udržuje tokotvorný (budící) proud konstantní, tím pádem máme k dispozici plný moment stroje – lepší dynamické vlastnosti. Při tomto řízení dochází k regulaci statorového toku v *dq* souřadnicích. Princip řízení je založen na řízení magnetického pole rotoru a statoru tak, aby byly mezi sebou vzájemně kolmé a nezávisle řiditelné.

Řízení dělíme na přímé a nepřímé. Přímé vektorové řízení využívá informaci o poloze vektoru magnetického toku nebo z výpočtů pomocí matematického modelu stroje. U nepřímého vektorového řízení se využívají pouze měřené hodnoty napětí a proudů.

Při regulaci jsou fázové proudy transformovány do souřadného systému *dq* pomocí Clarkové a Parkovy transformace. Výstupem regulace jsou napětí, která pomocí inverzních transformací převedeme na střídavé složky. V mikroprocesoru provádí inverzní Clarkové transformaci algoritmus pro generování PWM modulace.

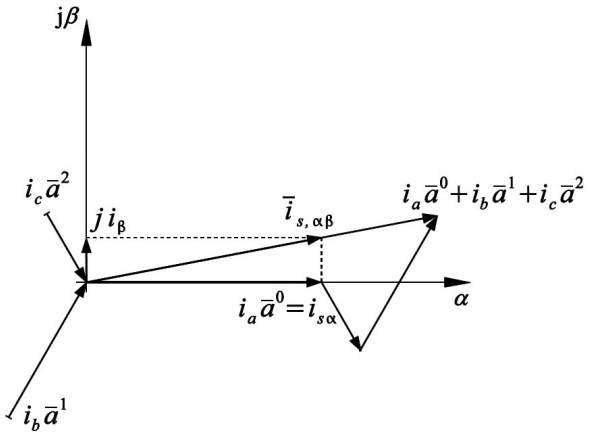
Transformaci veličin do *dq* souřadného systému vyžaduje vysoký výpočetní výkon mikrokontroleru, proto v našem případě využíváme digitální signálový procesor (DSP) [3; 9; 10].



Obrázek 11 Schéma vektorově orientovaného řízení asynchronního stroje – převzato z [10]

4.3 Algoritmy v řízení střídavých strojů

Pro popis účinků stavových veličin v elektrickém stroji lze zavést prostorové vektory, které využíváme pro odvození náhradního dvojfázového modelu stroje. Tyto modely jsou implementovány v mikroprocesorovém řízení střídavých pohonů. Prostorový vektor je popisován v souřadném systému $\alpha\beta$, kde tento souřadný systém je pevně svázán se statorem. Pro vektorové řízení převádíme vektory rotorových a statorových proudů $\bar{\imath}_R, \bar{\imath}_S$, vektorů rotorových a statorových napětí $\bar{u}_R, \bar{u}_S$ a vektorů rotorových a statorových magnetických toků $\bar{\psi}_R, \bar{\psi}_S$ do souřadného systému $\alpha\beta$, kde jsou jednotlivé složky prostorových vektorů $\bar{\imath}_{S\alpha}, \bar{\imath}_{S\beta}, \bar{u}_{S\alpha}, \bar{u}_{S\beta}, \bar{\psi}_{S\alpha}$ a $\bar{\psi}_{S\beta}$ [10].



Obrázek 12 Konstrukce prostorového vektoru statorového proudu – převzato z [10].

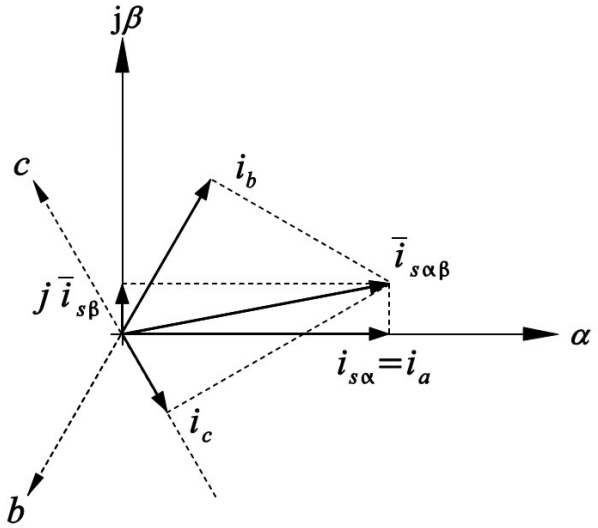
4.3.1 Clarkové transformace

Clarkové transformace je převod veličin ze 3 na 2. V našem případě je to převod fázových veličin ve stacionárním systému $\alpha\beta$ na jednotlivé složky prostorového vektoru. Výsledná velikost vektoru je zmenšená na 2/3 tak, aby jeho amplituda souhlasila s amplitudou v jedné fázi – transformace je amplitudově invariantní. Jednotlivé složky prostorového vektoru:

$$i_{s\alpha} = \frac{1}{3} \cdot (2i_a - i_b - i_c) = i_a \qquad (1)$$

$$i_{s\beta} = \frac{1}{\sqrt{3}} \cdot (i_c - i_b) \qquad (2)$$

Převod ze dvou na tři veličiny se nazývá inverzní Clarkové transformací, která znamená, že se jedná o průmět prostorového vektoru do os jednotlivých statorových cívek [10].

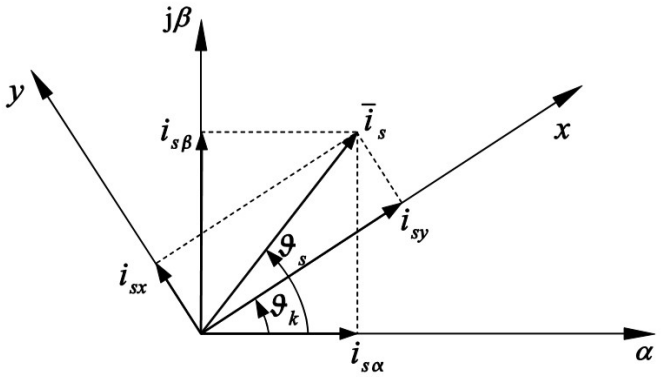


Obrázek 13 Inverzní Clarkové transformace – převzato z [10].

4.3.2 Parkova transformace

Parkova transformace se využívá u vektorového řízení, jelikož je souřadný systém spřažený s prostorovým vektorem některého z magnetických toků. V případě spřažení souřadného systému se spřaženým rotorovým tokem je potřeba souřadný systém pootočit o úhel υ_K [10]. Poloha natočení v souřadném systému je dána:

$$\bar{\imath}_{s,xy} = \bar{\imath}_{s,\alpha\beta} e^{-j\upsilon_K} \quad (3)$$



Obrázek 14 Parkova transformace – převzato z [10].

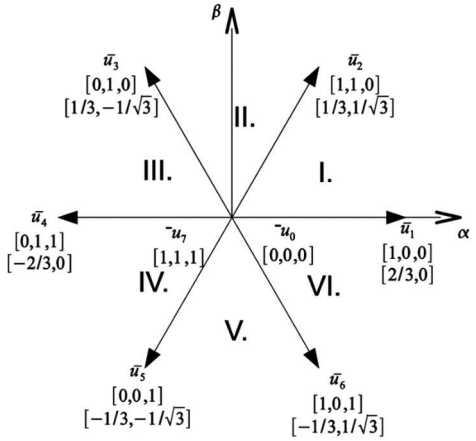
Pro inverzní Parkovu transformaci platí vztah:

$$\bar{\imath}_{s,\alpha\beta} = \bar{\imath}_{s,xy} e^{j\upsilon_K} \quad (4)$$

4.4 Modulace prostorového vektoru napětí (SVM)

Nevýhodou klasické pulsní šířkové modulace je to, že výstupní napětí může pouze dosáhnout 86 % efektivní hodnoty síťového napětí [3].

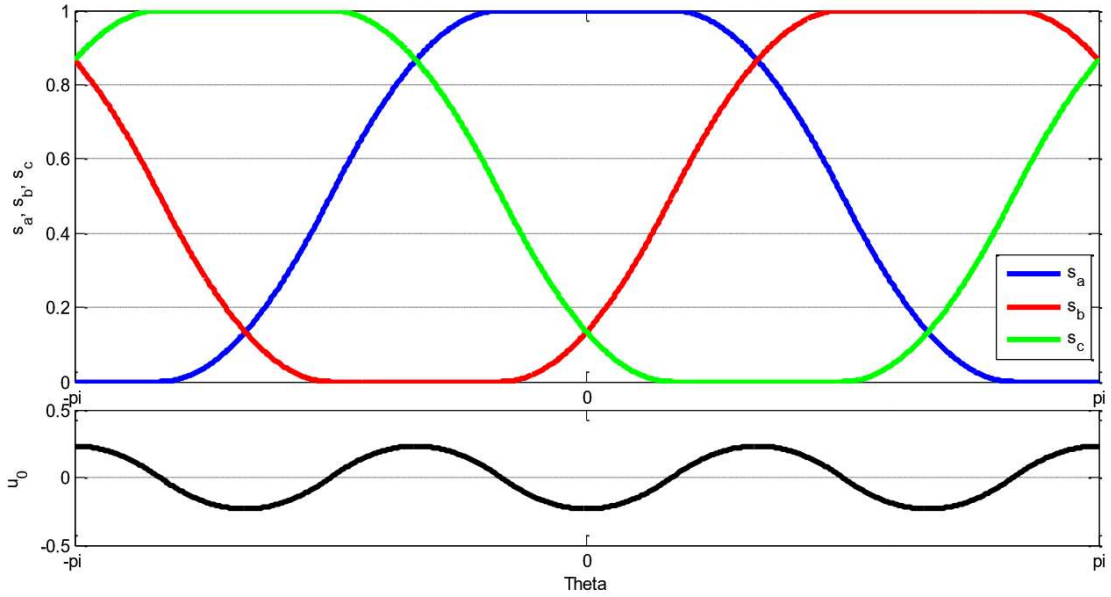
Pokročilejší metodou je modulace prostorového vektoru napětí, která odstraňuje nedostatky PWM. Při řízení motoru pomocí SVM lze dosáhnout efektivní hodnoty síťového napětí. Jelikož má střídač celkem tři větve, tak existuje celkem osm provozních stavů, ve kterých může střídač pracovat. Z osmi stavů je celkem šest aktivních stavů ***U1*** až ***U6*** a dva nulové stavy ***U0*** a ***U7*** [3].



Obrázek 15 Modulace prostorového vektoru – převzato z [10].

Potřebnou hodnotu napětí vektoru U_s získáme pomocí algoritmu SVM, která funguje tak, že jednotlivé vektory přepíná mezi sebou v určitém časovém sledu. Součet všech časových sledů je roven vzorkovací periodě PWM modulace.

Maximální amplitudu výstupního napětí střídače je možné dosáhnout tak, že střídač pracuje v *six-step* režimu. Modulační činitel je poměr prvního harmonického sdruženého napětí a napětí stejnosměrného napěťového meziobvodu [10]. V našem případě je při jmenovité frekvenci motoru modulační činitel roven jedné. Ve firmwaru je zvolen modifikovaný algoritmus SVM s vychylovacím napětím s injektovanými sinusovými vrchlíky. Injektované sinusové vrchlíky v uzlu statorového vinutí mají průběh třetí harmonické.



Obrázek 16 Průběh stříd jednotlivých větví a vychylovacího napětí při algoritmu PWM s injektováním sinusových vrchlíků a hloubce modulace M = 1 - převzato z [10].

5 FIRMWARE PRO SIGNÁLOVÝ MIKROKONTROLER MC56F82748

Firmware pro mikrokontroler je napsaný v jazyce C a v Assembleru. V programu jsou použity následující knihovny z FSLESL (FreeScale Embedded Software Libraries):

- knihovna *aclib.h* [11] -výpočet přetížení motorů (integrace nadproudu), integrace úhlové rychlosti.
- knihovna *mlib.h* [12] složitější matematické výpočty.
- knihovna *mclib.h* [13] -Clarkové transformace, inverzní Parková transformace a SVM modulace.
- knihovna *gflib.h* [14] -výpočet rampy, složek statorového toku a výpočet druhé odmocniny.

5.1 Řídící firmware pro stejnosměrný motor

Ve zjednodušeném vývojovém diagramu (Obrázek 33) je vidět základní schéma funkce programu pro stejnosměrný motor.

5.1.1 Hlavní program

Při zapnutí mikrokontroleru se nejprve firmware zinicializuje. Inicializace probíhá tak, že se nejdříve zahrnou systémové knihovny, následně FSLESL a další knihovny.
FSLESL knihovny:

```
/* FSLESL libraries */
#include "gflib.h"
#include "aclib.h"
#include "mclib.h"
#include "mlib.h"
```

Mezi další knihovny patří knihovny:

```
/* Hardware libraries */
#include "processor_pins.h"
/* Calculation libraries */
#include "common_types_mscan.h"
/* Regulator */
#include "pidparallel.h"
```

V první knihovně jsou definovány I/O porty, ve druhé knihovně jsou definovány společné datové typy a třetí knihovna je napsaná v jazyce Assembler a slouží k nastavení a výpočtu paralelního PID regulátoru [3]. Dále jsou v programu definovány globální proměnné, základní výpočty potřebné pro běh firmwaru, následuje inicializace periférií, které využíváme, a povolení všech potřebných přerušení.

Nyní následuje běh programu v nekonečné smyčce. Při vzniku přerušení od jakékoliv periférie se provede daná obslužná funkce. Nejvyšší prioritu (Level 2) má přerušení obslužného programu *PWM_fault*, který při signálu z budičů – zapůsobení saturační ochrany – vypne celý měnič, z důvodu ochrany výkonových tranzistorů. V našem případě je toto přerušení zakázané, protože při testování programu

docházelo k vypínání měniče z důvodu chybného signálu z budičů – špatně nastavené saturační napětí. Střední prioritu (Level 1) má přerušení A/D převodníku, které je vyvoláno signálem od komparátoru, který porovnává hodnotu 16-ti bitového čítače a hodnoty registru VAL0. Nejnižší prioritu přerušení (Level 0) má periodický časovač PIT0 a obsluhy přerušení komunikace MSCAN – CAN Error, CAN Receive Warning, Can Transmit Warning a CAN Wake-up.

V hlavním programu je funkce `FMSTR_Poll()`, která slouží k posílání dat do programu FreeMASTER a musí být periodicky volána v hlavní smyčce programu. Dále se v hlavním programu vyhodnocuje proměnná `StandStat.faultClear`, která je posílána do MCU z programu FreeMASTER při zapínání měniče, který mohl být před tím vypnutý nebo mohla nastat specifikovaná chyba – například došlo k přetížení motoru. Další proměnná je `StandStat.enable`, která nám určuje stav měniče. Jestliže daná proměnná má hodnotu *true*, tak je měnič zapnutý. V případě že má hodnotu *false*, tak se vypnou výstupy PWM z procesoru, rozsvítí se signalizační dioda a odešle se zpráva o stavu do druhého měniče – druhý měnič se též vypne.

Dále se v hlavním programu zpracovávají tři softwarové časovače:

- v prvním softwarovém časovači `sPIT0` se periodicky přepočítávají parametry PID regulátoru proudu kotvou, otáček a polohy rotoru.
- v druhém softwarovém časovači `sPIT1` se periodicky odesílají zprávy vyšší priority *msgHiPriority* do druhého procesoru s periodou 50ms.
- V třetím softwarovým časovači `sPIT2` se periodicky odesílají zprávy nižší priority *msgLoPriority* do druhého procesoru s periodou 100ms s počátečním zpožděním 25ms (aby se nikdy nemohly odesílat zprávy najednou).

Odesílání zprávy se provádí tak, že se zakáže přerušení pro příjem zprávy. Následně se zjistí, který odesílací buffer je volný, získá se ukazatel na paměťové místo odesílacího bufferu a následně do odesílacího bufferu uložíme 8 bajtů odesílané zprávy. Nyní se povolí přerušení pro přijímání dat pomocí sběrnice CAN, nastaví se délka zprávy na 8 bajtů, nastaví se ID odesílané zprávy, a odešle se zpráva z daného odesílacího bufferu (*txbuff*). Zprávy se odesílají jak periodicky, tak i při vzniku události (vypnutí měniče).

5.1.2 Obsluha přerušení při vybavení saturační ochrany

Přerušení je vyvolané signálem z budičů v případě, že se zvýší saturační napětí nad 1V, což odpovídá proudu 100A protékající výkonovým tranzistorem MOS-FET.

V obslužném programu přerušení se změní hodnota proměnné `StandStat.enable` na *false*, zakáže se výstup PWM a nastaví se digitální výstup LED na hodnotu *true* pomocí funkce `ioctl(GPIO_LED, GPIO_SET_PIN, LED)`. Následně se vymaže příznak přerušení `ioctl(EFPWMA, EFPWM_CLEAR_FAULT_FLAGS, EFPWM_FAULT_0)` a nastaví se proměnná `dataToSend.msgHiPriority.msgHi.StandStat.SaturationProtection` a `StandStat.SaturationProtection` na hodnotu *true*. Tyto dvě proměnné informují uživatele ve vizualizačním prostředí, že došlo k vybavení saturační ochrany výkonového modulu s MOS-FET tranzistory.

Funkce *ioctl()* usnadňuje přístup k periferiím, též slouží k inicializaci periferií. Hlavním důvodem používání této funkce je to, že se zvyšuje přenositelnost a srozumitelnost kódu.

5.1.3 Obsluha přerušení periodického časovače

V periodickém časovači PIT0 s periodou přerušení T = 1ms se provádí testování, zda jsou budiče hardwarovým přepínačem zapnuty, v případě že ano, tak se nastaví proměnná `StandStat.enableSwitch` na hodnotu *true*, v opačném případě na hodnotu *false*. Následuje načtení polohy rotoru a výpočet rychlosti otáčení rotoru, výpočet regulátoru otáček a polohy a výpočet softwarových časovačů. Název obslužné funkce je *pit0_isr*.

Zdroj informace o poloze rotoru je získávána pomocí kvadraturního čítače z emulovaného inkrementálního enkodéru obvodu AD2S1200, který získává informaci o poloze z resolveru, který je umístěný na hřídeli indukčního motoru.

Dále v obsluze přerušení je výpočet regulátoru pozice a otáček rotoru. Daný regulátor je povolený v závislosti na stavu proměnných `StandStat.PositionRegEnable` a `StandStat.SpeedRegEnable`. Na následujícím kódu je vidět podmínka, že pokud je regulátor otáček povolen, tak se periodicky vypočítává regulátor. V případě, že regulátor otáček je zakázaný, tak se nuluje regulační odchylka PID regulátoru.

```
if(StandStat.SpeedRegEnable)
{
      regData.DCcurrentDesired = pidParallel (SpeedDesired,
      Speed, &SpeedPidData);
}
else{
      SpeedPidData.yk1 =0;
      SpeedPidData.ek1 =0;
}
```

Dále obslužná funkce obsahuje softwarové čítače `sPIT0`, `sPIT1` a `sPIT2`. Princip softwarového časovače je takový, že se periodicky inkrementuje proměnná `sPIT0.counter`, následně se hodnota proměnné porovnává s hodnotou referenční – proměnná `sPIT0.period`, v případě splnění podmínky se následně nastaví proměnná `sPIT0.counter` do nuly a proměnná `sPIT0.run` se nastaví na hodnotu *true*.

Ukázka jednoho z časovačů:

```
sPIT0.counter++;
if(sPIT0.counter > sPIT0.period){
      sPIT0.counter = 0;
      sPIT0.run = 1;
}
```

Následuje vynulování příznaku přerušení

```
ioctl(PIT_0, PIT_CLEAR_ROLLOVER_INT, NULL).
```

5.1.4 Obsluha přerušení po přijmutí zprávy

Vyvoláním podprogramu (obsluhy přerušení) se získá *ID*, délka zprávy. Následuje testování, zda se zpráva přijme pomocí *ID* zprávy. V případě, že *ID* zprávy odpovídá požadovaným *ID*, tak se následně získá ukazatel na přijímací buffer, kde jsou data uložena. Následuje přijetí zprávy o délce 8 bajtů.

```
register UWord16* pdRHi = ioctl(MSCAN_RB, MSCANMB_GET_DATAPTR,
NULL);

if(pdRHi[0] == recData.msgHiPriority.msgHi.msgCode){
recData.msgHiPriority.message_template.msgCode = pdRHi[0];
recData.msgHiPriority.message_template.data[0] = pdRHi[1];
recData.msgHiPriority.message_template.data[1] = pdRHi[2];
recData.msgHiPriority.message_template.data[2] = pdRHi[3];
recData.msgHiPriority.message_template.data[3] = pdRHi[4];
recData.msgHiPriority.message_template.data[4] = pdRHi[5];
recData.msgHiPriority.message_template.data[5] = pdRHi[6];
recData.msgHiPriority.message_template.data[6] = pdRHi[7];
}
```

Datová struktury komunikace:

```
typedef struct{
      UWord8      msgCode;
      UWord8      data[7];
}t_message_template;

typedef struct{
      UWord8      msgCode;
      t_StandStat StandStat;
      t_ResBits   ResolverBits;
      Frac16      Speed;
      UWord8      reserved[2];
}t_msgHi;

typedef struct{
      UWord8      msgCode;
      Frac16      AMStatorFreqDesired;
      Frac16      AMSetSkluz;
      Frac16      AMcurrent;
      UWord8      reserved[1];
}t_msgLo;
typedef struct{
      union{
            t_message_template message_template;
            t_msgHi msgHi;
      }msgHiPriority;
      union{
            t_message_template message_template;
            t_msgLo msgLo;
      }msgLoPriority;
}t_mscan;
```

Tabulka 3 Popis proměnných v datové struktuře t_msgLo a t_msgHi

msgCode	Kód zprávy
standStat	Datová struktura stavu měniče
ResolverBits	Datová struktura se stavovými proměnnými resolveru
Speed	Rychlost otáčení hřídele motoru
AMStatorFreqDesired	Požadovaná hodnota statorového frekvence indukčního motoru
AMSetSkluz	Požadovaná hodnota skluzu motoru
AMcurrent	Absolutní hodnota proudu indukčního motoru

```
typedef struct{
      UWord8 PAR:1;
      UWord8 LOT:1;
      UWord8 DOS:1;
      UWord8 RDVEL:1;
      UWord8 reserved:4;
}t_ResBits;
```

Datová struktura *t_ResBits* slouží k informaci uživatele o ztrátě či porušení dat přijatých přes SPI z R/D převodníku.

```
typedef struct{
      UWord16 enable:1;
      UWord16 faultClear:1;
      UWord16 SpeedRegEnable:1;
      UWord16 PositionRegEnable:1;
      UWord16 CurrentRegEnable:1;
      UWord16 HWEnable:1; //nepoužito
      UWord16 AMmotorNorm:1; //nepoužito
      UWord16 AMmotorDynBreak:1;
      UWord16 StandStatEnAM:1;
      UWord16 MasterSlave:1;  //0 pro master, 1 pro slave
      UWord16 CurrentOverload:1;
      UWord16 SaturationProtection:1;
      UWord16 enableSwitch:1;
      UWord16 reserved:3;
}t_StandStat;
```

Datová struktura *t_StandStat* slouží k předávání stavu mezi procesory a uživatelským prostředím.

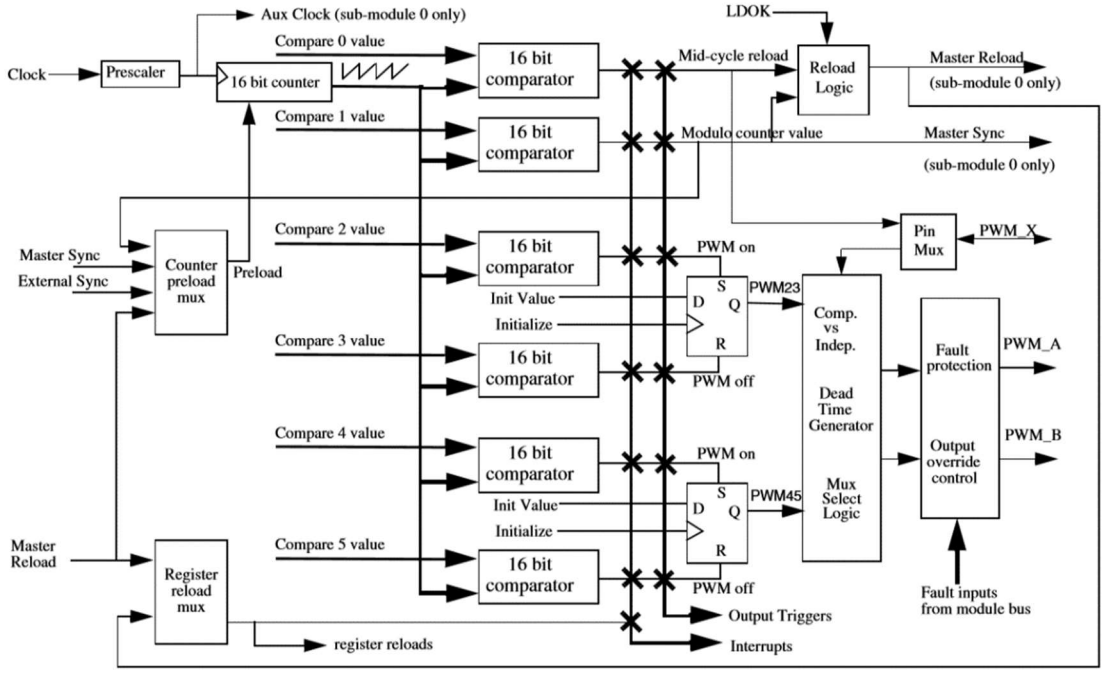
Nejdůležitější proměnou v této datové struktuře je proměnná *enable*, která nám určuje, jestli je měnič zapnutý nebo vypnutý. Další důležitá proměnná je *faultClear*, která povoluje znovu zapnutí měniče. Proměnné *SpeedRerEnable*, *PositionRegEnable* a *CurrentRegEnable* slouží k zapínání a vypínání jednotlivých regulátorů. Proměnné *CurrentOverload* a *SaturationProtection* nám říkají, jaká porucha nastala při vypnutí měniče. Proměnnou *AMmotorDynBreak* dáváme měniči informaci, jestli se jedná o provoz indukčního motoru v režimu skalárního řízení nebo brždění pomocí změny skluzu. Proměnná *StandStatEnAM* je pomocnou proměnnou pro *FreeMASTER*, tato proměnná nám říká, zda je indukční motor povolen, či není. Poslední proměnnou

v datové struktuře je proměnná *enableSwitch*, která nám říká, zda jsou zapnuté budiče, pokud je hodnota této proměnné *false*, tak se oba měniče vypnou.

Ostatní obsluhy přerušení (*MSCAN_ErrorISR, MSCAN_WakeUpISR, MSCAN_TxRdyISR*) od periferie MSCAN se pro naše účely nevyužívají.

5.1.5 Nastavení pulsní šířkové modulace

Jak bylo již napsáno, tak jsme zvolili unipolární řízení stejnosměrného motoru s frekvencí PWM modulace f = 10kHz. Dále je nastaven DeadTime pro tranzistory na hodnotu 1us a je nastaven trigger na porovnání hodnoty VAL0 a 16-ti bitového čítače - na blokovém schématu PWM (Obrázek 17) jsou vidět oba registry, které se porovnávají.



Obrázek 17 PWM blokové schéma – převzato z [15].

5.1.6 Obsluha přerušení analogového převodníku

V obsluze přerušení A/D převodníku se provádí načtení hodnot napětí a proudů, výpočet odchylky proudu proudového čidla (nulování offsetu), výpočet regulátoru proudu, výpočet přetížení motoru, nastavují se střídy PWM modulace a ukládají se data do bufferu pro zobrazení měřených hodnot do *FreeMASTERu*.

Při vzniku příznaku přerušení od komparátoru čítače a registru VAL0 u PWM periférie nastane nejprve přečtení hodnoty napětí a proudu.

```
      regData.DCcurrent = (ioctl(ADC, ADC_READ_SAMPLE, 0) -
CurrentOffset);
      regData.DCVoltage = ioctl(ADC, ADC_READ_SAMPLE, 1);
```

Jelikož proudové čidlo má na svém výstupu určitý offset, tak musí být do firmwaru implementován kód, který nuluje offset čidla. Tento kód se spouští pouze po zapnutí standu, nebo při resetování MCU a celkem se opakuje pro vyšší přesnost 1000krát.

```
if(CurrentOffsetCompenzation)
{
      if(sample < 1000)
      {
            CurrentSample += regData.DCcurrent/10;
      }else
            {
                  CurrentOffset = CurrentSample/100;
                  CurrentOffsetCompenzation = false;
            }
      sample++;
}
```

Nyní podle proměnné `StandStat.CurrentRegEnable` se určí, jestli se bude provádět výpočet střídy pomocí PID regulátoru nebo se bude střída zadávat ručně ve vizualizačním prostředí.

```
if(StandStat.CurrentRegEnable){
      regData.DCduty = pidParallel (regData.DCcurrentDesired,
      regData.DCcurrent, &CurrPidData);
}else
{
      regData.DCduty = GFLIB_Ramp16(DCDutyDesired,
      regData.DCduty,&sDCDutyParam);
      CurrPidData.yk1 = 0;
      CurrPidData.ek1 = 0;
}
```

V případě, že bude proměnná `StandStat.CurrentRegEnable` mít hodnotu *false*, tak se změní střída podle zadání uživatele ve vizualizačním prostředí. Změna nebude provedena skokem, ale rampou.

Pro výpočet přetížení je potřeba integrovat nadproud (proměnná `eABScurr`), který protéká kotvou stejnosměrného motoru. Nadproud se vypočítá jako absolutní hodnota aktuálního proudu mínus jmenovitá hodnota proudu.

```
eABScurr = MLIB_Abs16(regData.DCcurrent)- FRAC16(0.1245405405);
```

Následně se porovnává, jestli je hodnota rozdílu proudů kladná nebo záporná. V případě, že je rozdíl kladný, tak se začne nadproud integrovat. V případě, že je rozdíl záporný, integrátor začne integrovat směrem dolů v závislosti na softwarovém časovači `sPIT2`.

```
if(((eABScurr > 0) || (OverCurrent > 0))&&sPIT2.run){
      sPIT2.run = 0;
      OverCurrent =
      ACLIB_Integrator(eABScurr,OverCurrentIntegrator);
}
```

Při snížení nadproudu pod nulovou hodnotu se nastaví počáteční hodnota integrátoru na 0. Nadproudová ochrana má celkem dva prvky.

- První prvek integruje nadproud
- Druhý prvek je mžiková ochrana motoru při zvýšení proudu nad 46,96A

V případě vybavení ochrany motoru se oba měniče vypnou a předají informaci do vizualizačního prostředí s informací, že zapůsobila nadproudová ochrana stejnosměrného motoru.

V dalším kroku se nahraje požadovaná střída do PWM modulátoru a uloží se data do bufferu.

5.2 Řídící firmware pro indukční motor

Ve zjednodušeném vývojovém diagramu (Obrázek 34Obrázek 33) je vidět základní schéma funkce programu pro indukční motor.

5.2.1 Hlavní program

Průběh hlavního programu je obdobný jako u měniče pro stejnosměrný motor (kapitola 5.1.1). Nejprve se zinicializují jednotlivé periférie, povolí se přerušení a počká se na povel zapnutí měniče. Zkontroluje se stav měniče, zda nenastal nějaký podnět k vypnutí, dále následují periodické výpočty a odeslání zprávy do druhého měniče v případě vypnutí poruchy.

Jediná odlišnost hlavního programu měniče pro indukční motor je ten, že se změnil počet softwarových časovačů a úkonů, které se periodicky provádějí. Dále se ještě zahrnují další čtyři knihovny [3].

```
#include "am_scalar.h"
#include "am_setup.h"
#include "uf_calc.h"
#include "velocity.h"
```

V první knihovně je definovaná funkce rampy, u které je rychlejší výpočet něž u funkce *GFLIB_Ramp16()*, dále knihovna obsahuje funkci, ve které se inicializuje výpočet SVM – *init_svm_calc()*. Nastaví se souřadnice α, β a D, Q do počátku, složky úhlu pootočení (sin =0 a cos =1), vynuluje se počáteční hodnota integrátoru a nastaví se jeho zesílení a shift. Poslední funkcí je funkce pro získání střídy ze statorové frekvence *getPwmDuty()*. Ve funkci se nejprve integruje statorová frekvence, ze které dostaneme po integraci statorový tok, následně se vypočítává složka sinus a cosinus ze statorového toku pomocí aproximace Taylorova polynomu devátého řádu. Dále se pomocí inverzní Parkovy transformace získá složka α, β, ze kterých se pomocí funkce *MCLIB_SvmSci()* získají hodnoty střídy PWM modulátorů.

Druhá knihovna slouží k nastavení parametrů měniče – maximální otáčky, jmenovitá frekvence motoru, minimální frekvence, minimální napětí *U/f* řízení,… V třetí knihovně se vypočítává napětí pro *U/f* řízení. V poslední knihovně se vypočítává rychlost rotoru z rozdílu polohy rotoru s frekvencí $f_{PWM}/10$.

Firmware pro indukční motor obsahuje celkem dva softwarové časovače. První z nich *sPIT0* periodicky čte hodnotu napětí z AD převodníku, dále se přenastavují hodnoty jednotlivých proměnných podle hodnot přijatých proměnných z druhého měniče a počítá se nulová složka absolutní hodnoty proudu s periodou T =50ms. Druhý softwarový časovač *sPIT1 j*e určen pro počítání rampy statorové frekvence proudu a výpočet nadproudu s periodou T = 10ms.

5.2.2 Obsluha přerušení při vybavení saturační ochrany

Obsluha přerušení je obdobná obsluze přerušení u stejnosměrného motoru. Jediný rozdíl je ten, že se vypínají tři PWM kanály místo dvou. Popis obsluhy přerušení lze nalézt v kapitole 5.1.2.
Ukázka kódu obsluhy přerušení:

```
void pwm_fault(void)
{
      ioctl(EFPWM_A, EFPWM_CLEAR_FAULT_FLAGS, BIT_0);
      ioctl(EFPWMA, EFPWM_SET_OUTPUTS_DISABLE, PWM_3F_ALL);
      StandStat.enable = 0;
      ioctl(GPIO_LED, GPIO_SET_PIN, LED);
      dataToSend.message.message.StandStat.SaturationProtection =
      1;
      StandStat.SaturationProtection = 1;
}
```

5.2.3 Obsluha přerušení periodického časovače

Obslužný program je volán s periodou T = 1ms. V těle obslužného programu se testuje stav hardwarového přepínače budičů, spouštějí se pouze dva softwarové časovače *sPIT* a maže se příznak přerušení pomocí funkce:

ioctl(PIT_0, PIT_CLEAR_ROLLOVER_INT, NULL).

Princip softwarového časovače je popsán v kapitole 5.1.3.

5.2.4 Obsluha přerušení po přijmutí zprávy

Vyvoláním podprogramu (obsluhy přerušení) se získá *ID*, délka zprávy. Následuje testování, zda se zpráva přijme pomocí *ID* zprávy. V případě, že *ID* zprávy odpovídá požadovaným *ID*, tak se zjistí pomocí funkce

```
txbuff = ioctl(MSCAN, MSCAN_SELECT_TXBUFF, MSCAN_ANY_TXBUFFER);
```

zda je volný nějaký z odesílacích bufferů, následně se získá ukazatel na přijímací buffer, kde jsou data uložena. Dále se hodnoty z přijímacího bufferu uloží do proměnných. Následuje kopírování odesílaných dat do odesílacího bufferu, nastaveni *ID* zprávy, délky zprávy a následně se zpráva odešle.

Detailní popis softwaru pro odesílání dat – viz kapitola 5.1.4.

5.2.5 Nastavení pulsní šířkové modulace

Během testování kódu bylo zjištěno, že při frekvenci PWM modulace f_{PWM} = 20kHz není dostatek času pro spolehlivý chod měniče, jelikož v obsluze přerušení

analogového převodníku je mnoho událostí, které jsou dost časově náročné. Proto byla zvolena frekvence PWM modulace f_{PWM} = 10kHz.

Při této frekvenci je větší časová rezerva pro všechny důležité výpočty. Dále je nastaven DeadTime pro tranzistory na hodnotu 1,5us a je nastaven trigger na porovnání hodnoty VAL0 a 16-ti bitového čítače - na blokovém schématu PWM (Obrázek 17) jsou vidět oba registry, které se porovnávají. Rozlišení PWM modulace je 2^{13}.

5.2.6 Obsluha přerušení analogového přerušení

V obsluze přerušení nejprve dojde k načtení otáček z R/D převodníku, dále se načtou hodnoty proudů ze dvou proudových senzorů, vypočte se absolutní hodnota proudu, přetížení motoru, rychlost otáčení rotoru, napětí *U/f* charakteristiky, výpočet střídy jednotlivých PWM modulátorů a pošlou se informace do FreeMASTERu.

Při vzniku příznaku přerušení od komparátoru čítače a registru VAL0 u PWM periférie nastane nejprve přečtení hodnoty proudů ve větvi A a C. Následně se proud větví B vypočte.

```
regData.currents.f16A = ioctl(ADC, ADC_READ_SAMPLE, 1);
regData.currents.f16C = ioctl(ADC, ADC_READ_SAMPLE, 0);
regData.currents.f16B = -regData.currents.f16C-
regData.currents.f16A;
```

Dále se přečte hodnota pozice rotoru a informace o stavu přenesených dat z obvodu AD2S1200 přes SPI.

```
regData.ResPosition = -(ioctl(SPI_0, SPI_READ_DATA, NULL) &
0xfff0);
regData.ResDiag.all = ioctl(SPI_0, SPI_READ_DATA, NULL) & 0x000f;
```

Proměnná `regData.ResDiag.all` obsahuje proměnnou *RDVEL*, která nám určuje, zda jsme obdrželi informaci o poloze nebo o otáčkách. Dále proměnná obsahuje bity *DOS*, *LOT* a *PAR*, které nám určují typ poruchy (poškození) informace z R/D převodníku. Hodnota *false* u bitu *DOS* znamená, že došlo k degradaci signálu a u bitu *LOS* znamená, že došlo k ztrátě sledování rotoru. V případě, že oba bity budou mít hodnotu *false*, došlo ke ztrátě signálu. Bit *PAR* nám reprezentuje paritu [4].

Výpočet absolutní hodnoty proudu je proveden pomocí Clarkové transformace, ze které dostaneme složky proudů *α, β*. Výsledná absolutní hodnota proudu se vypočte ze složek α, β pomocí Pithagorovy věty.

```
MCLIB_ClarkTrf(&regData.currentsAlfaBeta, &regData.currents );
dataToSend.message.message.AMcurrent =
GFLIB_SqrtPoly(L_mac(L_mult(regData.currentsAlfaBeta.f16Alpha,
regData.currentsAlfaBeta.f16Alpha),
regData.currentsAlfaBeta.f16Beta,
regData.currentsAlfaBeta.f16Beta));
```

Následuje výpočet nadproudu – rozdíl měřené hodnoty a jmenovité hodnoty proudu. Nadproud se integruje v softwarovým časovači `sPIT1` (viz kapitola 5.2.1 a 5.2.3). Dále se ověřuje hodnota nadproudu a aktuální absolutní hodnoty proudu s nastavenou maximální absolutní hodnotou proudu 45,89A.

Další částí programu je získání rychlosti otáčení rotoru z polohy rotoru pomocí funkce *getVelocity()*. Pro výpočet napětí z úhlové rychlosti rotoru je použita funkce *set_volatge()*. Před výpočtem napětí se testuje proměnná

`recData.message.message.StandStat.AMmotorDynBreak`.

Pokud má hodnotu *false,* tak se napětí vypočte ze statorové frekvence. Jestliže má hodnotu *true*, tak se napětí vypočte z přepočtené frekvence pomocí vztahu

$$f_{stat} = f_{otHř} * (1 - s) \quad (5)$$

, kde f_{stat} je frekvence statorového pole [-], $f_{itHř}$ je frekvence otáčení hřídele motoru [-] a s je požadovaný skluz motoru [-]. Frekvence jsou bez jednotky, jelikož se jedná o zlomkovou aritmetiku, tak musíme použít intrinsické funkce. Intrinsická funkce *L_mult()* násobí mezi sebou dvě 16 bitová čísla a výsledek uloží do 32 bitové frakční proměnné. Tato funkce nemohla být použita. Pro násobení byla použita funkce *mult().* Jelikož tato funkce má 16ti bitový výsledek, tak je možné mezi s sebou násobit čísla o maximálních hodnotách 0x8000 x 0x8000, v případě vyšších hodnot vstupních proměnných výsledek akumulátoru přeteče. Z tohoto důvodu je vztah roznásobený, abychom násobily nižší hodnoty frakčních proměnných mezi sebou.

```
if(recData.msgHiPriority.msgHi.StandStat.AMmotorDynBreak)
{
      regData.StatorFreq = regData.Speed - mult(regData.Speed,
      recData.msgLoPriority.msgLo.AMSetSkluz);
}

set_voltage(regData.StatorFreq, &regData.coorDQ.f16Q,
&regData.GainShift_Uf);
```

Nyní následuje získání střídy pomocí funkce *getPwmDuty()*. Dále se testuje podmínka, zda je statorová frekvence nulová. V případě, že je podmínka splněna, tak se nastaví střída PWM modulátorů na hodnotu 0,5. Tato podmínka byla vložena do firmwaru, jelikož při zastavení rotoru protékal proud statorovým vinutím.
Nahrání stříd do PWM modulátoru probíhá pomocí funkce:

```
ioctl(EFPWMA, EFPWM_CENTER_ALIGN_UPDATE_VALUE_REGS_COMPL_012,
&regData.duty);

ioctl(EFPWMA, EFPWM_SET_LOAD_OK, EFPWM_SUBMODULE_0
|EFPWM_SUBMODULE_1 | EFPWM_SUBMODULE_2);
```

Následně se nastaví samplovací bit pro R/D převodník a výstupní pin POS_SPEED. Nastavením hodnoty pinu na hodnotu *true*, převodník R/D pošle informaci o poloze, v případě, když je hodnota *false*, tak nám převodník pošle informaci o počtu otáček.

```
ioctl(GPIO_RESOLVER, GPIO_SET_PIN, RES_SAMPLE);
ioctl(GPIO_POS_SPEED, GPIO_SET_PIN, POS_SPEED);
```

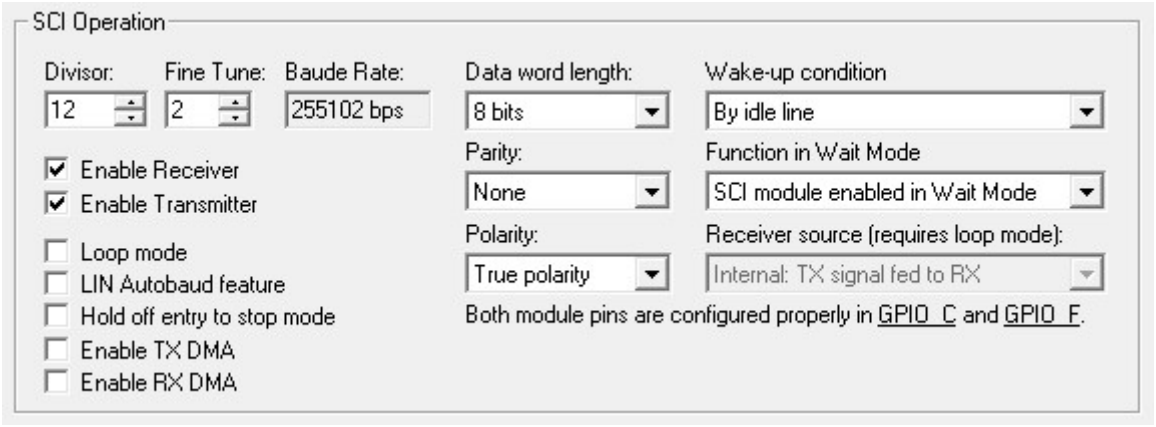
5.2.7 Obsluha přerušení periférie PWM

V této obsluze přerušení se jen řeší nulování výstupního bitu *RES_SAMPLE*. Tento bit výstupního portu sampluje polohu resolveru.

6 VIZUALIZAČNÍ PROSTŘEDÍ

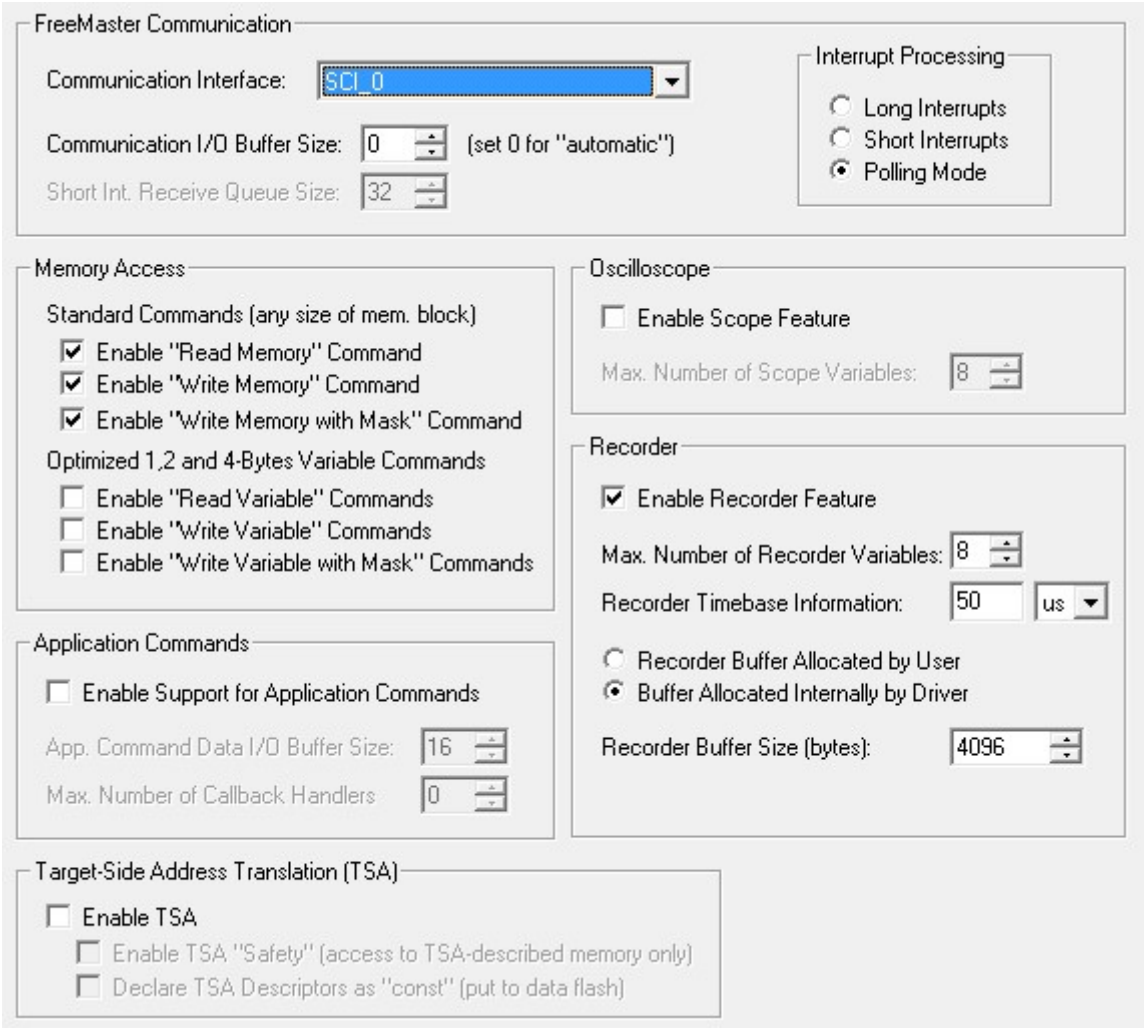
Vizualizační prostředí pro počítač slouží ke komunikaci se standem. Výrobce mikroprocesorů dává uživatelům možnost stáhnout si software FreeMASTER, který slouží k ladění firmwaru mikroprocesoru, sledování informací v reálném čase. Program umožňuje zobrazit průběhy jednotlivých proměnných v nástroji *Scope* v závislosti na čase nebo na jiné proměnné. Pro sledování proměnných, u kterých potřebujeme častější vzorkování, použijeme nástroj *Recorder*, který využívá místo v paměti mikroprocesoru. Hodnoty proměnných jsou ukládány do této paměti a po naplnění je obsah paměti odeslán do *Recorderu* ve FreeMASTERu [16].

V procesoru je nutné nastavit typ komunikace, jak bude mikrokontroler komunikovat s FreeMASTERem. V našem případě je na procesorové desce převodník SCI/USB. Tudíž zvolíme sériovou komunikaci SCI_0. Na obrázcích je vidět potřebné nastavení komunikace (Obrázek 18 a Obrázek 19).



Obrázek 18 Nastavení sériové komunikace pro FreeMASTER

Na následujícím obrázku je vidět, že jsme zvolili *Polling mód*. V tomto módu jsou zaznamenávána data a odesílána voláním funkce *FMSTR_Pool()*. Tato funkce se periodicky volá v nekonečné smyčce v hlavní programu mikrokontroleru. Pomocí funkce *FMSTR_Recorder()* se zaznamenávají hodnoty do paměti (bufferu). Tato funkce je umístěna v obsluze přerušení A/D převodníku.



Obrázek 19 Nastavení komunikace FreeMASTERu

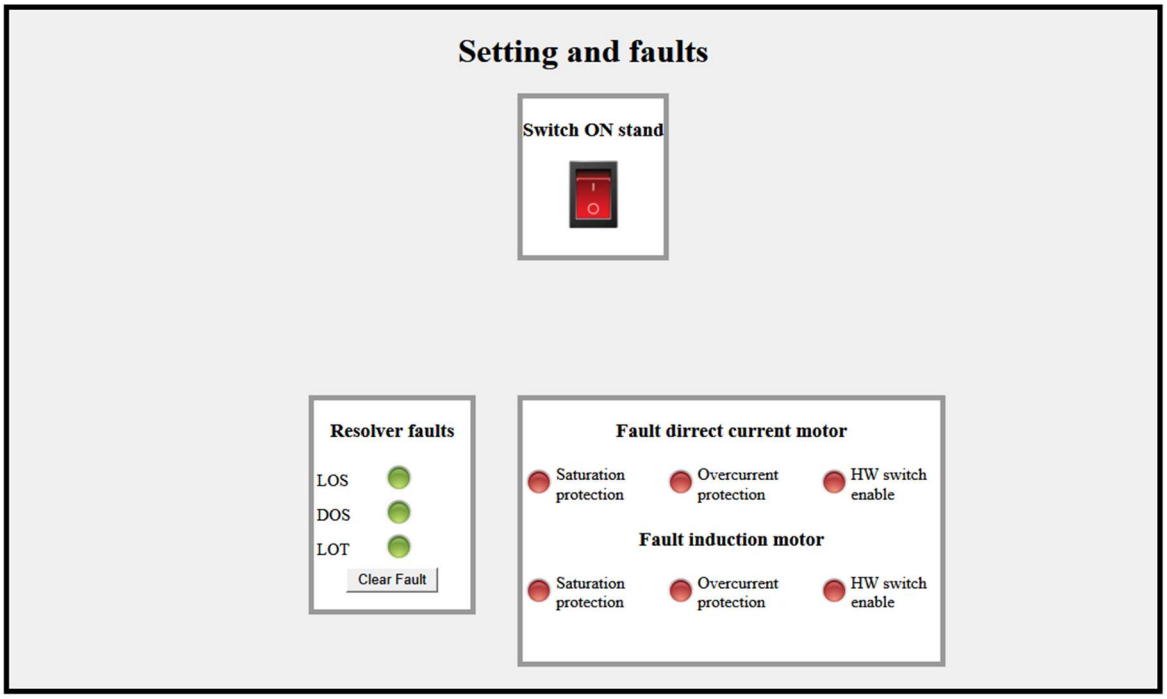
Jelikož ve firmwaru využíváme frakční proměnné, které mají rozsah <-1,1), musíme v uživatelském prostředí pro přehlednost proměnné přepočítávat pomocí určité směrnice, kterou si určíme tak, aby rozsah frakční proměnné odpovídal rozsahu měřené veličiny.

Pro lepší přehlednost jednotlivých proměnných je v této práci vytvořeno grafické vizualizační prostředí, které umožňuje nastavování jednotlivých regulátorů, motorů, nulování chyb atd.

Ve FreeMASTERu využíváme projektové menu, ve kterém přepínáme mezi jednotlivými okny, *Scopy* a *Recordery*. Následně využívaným oknem je okno *Variable Stimulus*, které nám slouží k nastavování hodnoty proměnné v závislosti na čase. Předposledním oknem je *Variable Watch*, kde vidíme potřebné hodnoty proměnných. Posledním oknem je grafické uživatelské prostředí, kterým se budeme v následujících kapitolách zabývat.

6.1 Úvodní obrazovka vizualizačního prostředí a reset chyb

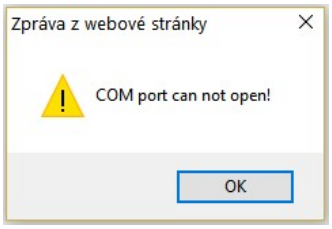
Úvodní okno grafického rozhraní slouží k zapínání, vypínání měničů a k resetování poruch. Další částí úvodního okna jsou informace o chybách. V případě, že se v panelu zobrazí chyba komunikace s resolverem, popřípadě přetížení jednoho z motorů, lze tlačítkem *Clear Fault* chybu vymazat a měnič se opět zapne. Grafické rozhraní úvodní stránky je vidět na následujícím obrázku.



Obrázek 20 Grafické rozhraní pro zapínání standu a nulování chyb

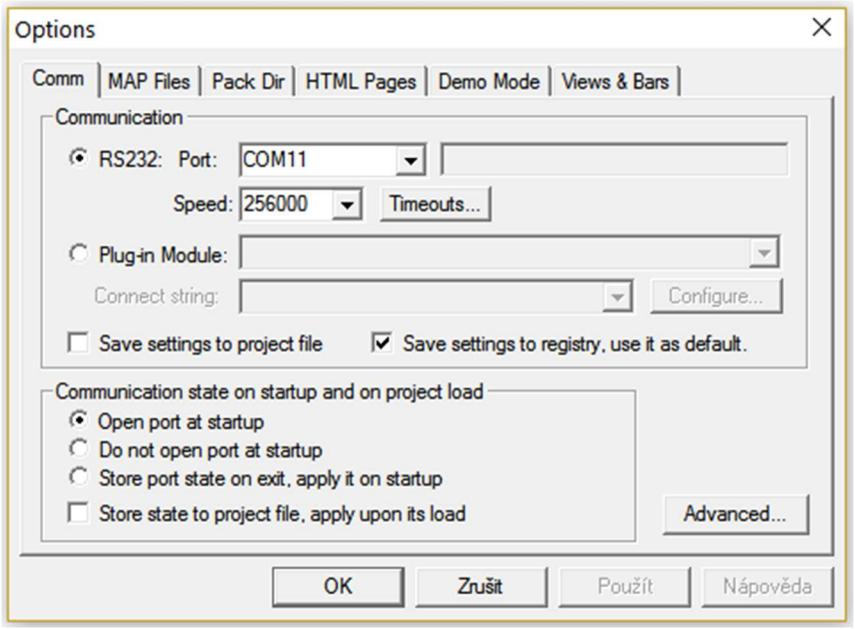
6.1.1 Popis jednotlivých prvků úvodní obrazovky

Po zapnutí vizualizačního prostředí a kliknutí na přepínač *Switch ON stand* se snaží software automaticky připojit k měniči stejnosměrného motoru. V případě, že program vyhodnotí chybu – nepřipojen kabel USB k měniči, špatná komunikační rychlost nebo zvolen nesprávný port – dojde k zobrazení chyby na obrazovce (Obrázek 21 - COM port nemůže být otevřen). Po kliknutí na tlačítko *OK* se znovu testuje, zda je měnič připojen. V případě, že měnič není připojen, se hláška zobrazí znovu a po opětovném kliknutí na tlačítko *OK* se uživatel dostane na úvodní grafické rozhraní, kde může následně přenastavit COM port nebo komunikační rychlost zařízení. Ve vizualizačním prostředí FreeMASTER lze v horní liště tlačítkem STOP otevřít komunikační port pro připojení měniče.



Obrázek 21 Chybová zpráva při zapínání měniče

Cesta k nastavení: *Project* → *Options…* Při rozkliknutí nabídky port v okně *Options* → *Comm* nám software nabídne porty, které jsou zrovna dostupné (připojené k počítači), vybereme správný komunikační port. Následně vybereme komunikační rychlost v nabídce *Speed*. Rychlost přenosu musí být nastavená na rychlost 256k baudu/s (Obrázek 22). Pro uložení nastavení – potvrdíme tlačítkem *OK*.



Obrázek 22 Okno s nastavením komunikace FreeMASTERu s mikrokontrolerem

Přepínačem *Switch ON stand* se stand zapíná a vypíná. V bloku *Resolver faults* jsou kontrolky, které zobrazují typ chyby resolveru. Popis chyb je vysvětlen v kapitole 5.2.6. Dále je v tomto bloku ještě tlačítko *Clear Fault,* pomocí kterého lze vymazat vzniklou chybu a zapnout oba měniče.

V posledním bloku vizualizace se zobrazují chyby vzniklé přetížením motorů, zapůsobením saturační ochrany a hardwarovým vypnutím budičů výkonového tranzistoru. Zelená kontrolka nám signalizuje stav systému bez poruchy. V případě poruchy se barva kontrolky změní ze zelené na červenou u dané poruchy a stand se vypne. V prvním řádku jsou chybové kontrolky pro stejnosměrný motor (*Fault dirrect current motor*). Ve druhém řádku jsou chybové kontrolky pro indukční motor (*Fault induction motor*). První sloupec bloku udává, že došlo k vybavení saturační ochrany výkonového modulu (*saturation protection*). Druhý sloupec udává, že došlo k poruše a vybavení ochrany z důvodu přetížení motorů (*overcurrent protection*).

Poslední sloupec udává, že došlo k vypnutí měniče z důvodu hardwarového vypnutí jednoho z bloku budičů.

6.1.2 Návrh grafického rozhraní úvodní obrazovky

Návrh grafického rozhraní je proveden formou webové stránky. Pro přesné umístění jednotlivých prvků na stránce byly použity kaskádové styly (CSS) a pro aktivní prvky je použit JavaScript. V HTML kódu se musí nejprve načíst objekt:

```
<OBJECT name="freemaster" id="fmstr" height="0" width="0" classid="
clsid:48A185F1-FFDB-11D3-80E3-00C04F176153"></OBJECT>
```

Bez načtení tohoto objektu by nebylo možné přečíst nebo zapsat hodnoty proměnných v mikrokontroleru, protože by nebyly dostupné volané funkce [17].

Po zapnutí se testuje, zda je měnič připojen.

```
if(fmstr.IsCommPortOpen()==false)
{
      fmstr.StartStopComm(true);
      Delay(3000);
      if(fmstr.IsCommPortOpen()==false) {
      window.alert("COM port can not open!");
      }
}
```

V případě, že je měnič připojen a zároveň je správně nastavená komunikace, počítač se připojí k měniči. Následně se volá funkce zpoždění, která počká tři sekundy a následně testuje, zda je měnič připojen. Pokud se měnič nepřipojí, zobrazí chybová hláška (Obrázek 21).

```
function Delay(milliseconds) {
  var start = new Date().getTime();
    for (var i = 0; i < 1e7; i++) {
      if ((new Date().getTime() - start) > milliseconds){
        break;
      }
    }
  }
```

Obnovování stavu kontrolek je nastaveno na dobu 500ms pomocí funkce *setInterval(function(){refresh_Indicators();},500)*. V případě, že uživatel klikne na tlačítko *Clear faults*, vyvolá následující funkce.

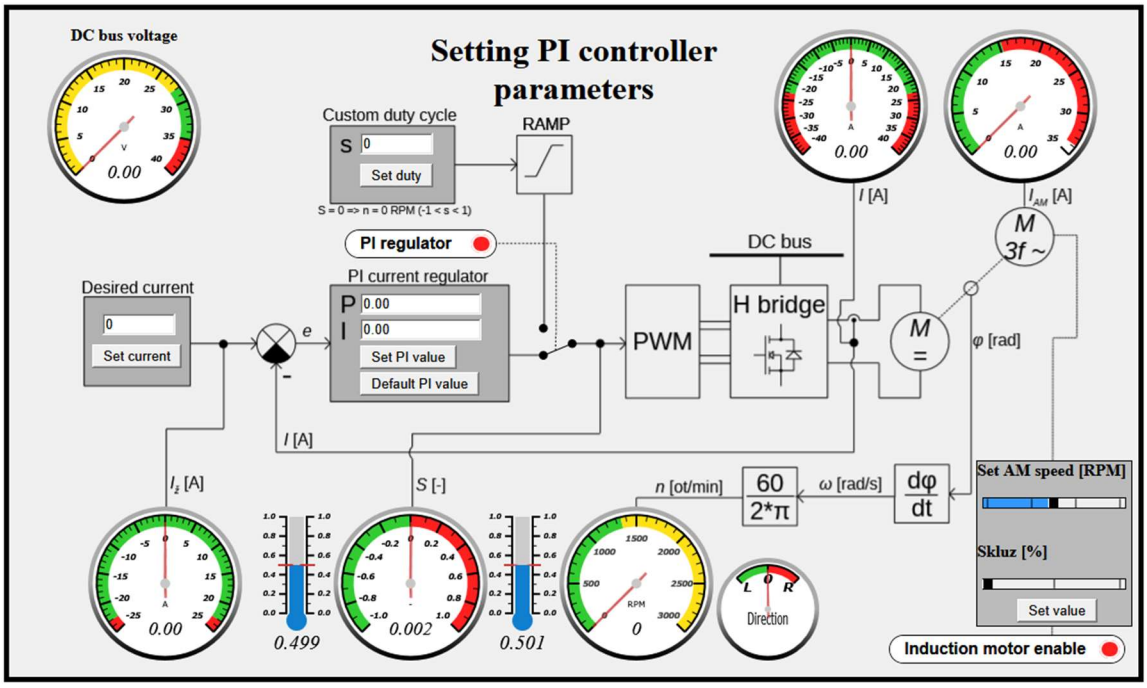
```
function clearFault() {
      fmstr.WriteVariable("StandStat.faultClear",1);
}
```

Pomocí funkce *fmstr.WriteVariable(variable, value)* změníme proměnnou *StandStat.faultClear* na hodnotu *true.* Pro čtení hodnoty proměnné musíme nejprve pomocí funkce *ReadVariable()* načíst hodnotu proměnné do položky *LastVariable_vValue*, jelikož JavaScript nepodporuje výstupní parametry funkcí. Funkci čtení proměnné si ukážeme na proměnné *StandStat.SaturationProtection*.

```
fmstr.ReadVariable("StandStat.SaturationProtection",
SaturationProtection);
SaturationProtection = fmstr.LastVariable_vValue;
```

6.2 První laboratorní úloha–návrh regulátoru proudu stejnosměrného motoru

V první laboratorní úloze budou studenti navrhovat PI regulátor proudu pomocí metody optimálního modulu. V měření této laboratorní úlohy se naučí, jakým způsobem se měří elektromagnetická časová konstanta τ_a, odpor a indukčnost vinutí kotvy stejnosměrného motoru, konstantu stroje $c\phi$. Následně pomocí změřených hodnot vypočítají PI regulátor proudu a ověří si, zda překmit proudu a rychlost náběhu je shodný s teorií.



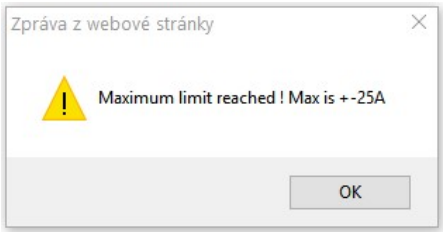
Obrázek 23 Grafické rozhraní první laboratorní úlohy

6.2.1 Popis jednotlivých prvků grafického rozhraní první laboratorní úlohu

Vzhled grafického prostředí je přizpůsoben studentům tak, aby co nejlépe pochopili zapojení a princip regulátoru proudu pro stejnosměrný motor. Blokové schéma umožňuje studentům zapnout, popřípadě vypnou regulátor proudu. V případě, že je regulátor proudu vypnut, je možné zadávat střídu přímo do PWM modulátoru. Zadaná střída se nezmění skokově, ale změní se pomocí rampy s pevně danou strmostí. Je-li zapnut PI regulátor proudu, jsou složky proporcionálního a integračního zesílení dány defaultně. Zapínání a vypínání regulátoru je prováděno aktivním přepínačem *PI regulator / Custom duty*. Student si vypočítá hodnoty jednotlivých složek regulátoru a klikne na tlačítko *Set PI value*. V bloku *Desired current* je možné zadat požadovanou hodnotu proudu a sledovat odezvu regulátoru.

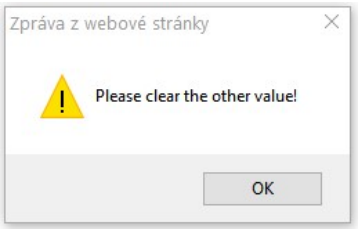
Následně je možnost odzkoušet regulátor, že správně reguluje proud při změnách zatížení pomoci indukčního motoru. Pro nastavení defaultních hodnot složek PI regulátoru stačí kliknout na tlačítko *Default PI value*. Dále vizualizační prostředí umožňuje zapnout indukční motor pomocí přepínače *Induction motor enable / Induction motor disable*. Nyní je možné roztočit indukční motor nebo nastavit skluz indukčního motoru. Nastavovaná rychlost indukčního motoru je synchronní rychlost statorového pole. Grafické rozhraní obsahuje celkem 9 aktivních grafických zobrazovačů [18], které zobrazují jednotlivé veličiny. Stupnice grafických zobrazovačů je přizpůsobena měřené veličině.

Ve vizualizačním prostředí jsou ošetřeny nežádoucí stavy. V případě, že student zadá vyšší hodnotu, než je rozsah dané veličiny, tak se zobrazí informační hláška, že hodnota je mimo rozsah. Příklad informační hlášky je na následujícím obrázku.



Obrázek 24 Informační hláška při překročení rozsahu.

V bloku nastavení indukčního motoru může být zvolen pouze jeden mód, a to nastavování rychlosti statorového pole nebo nastavování skluzu indukčního motoru (indukční motor slouží jako brzda). V případě, že se uživatel pokusí nastavit oba módy, tak se vypíše následující hláška.



Obrázek 25 Informační hláška indukčního motoru v případě špatného nastavení

6.2.2 Návrh grafického rozhraní první laboratorní úlohy

Na začátku kódu je definované umístění jednotlivých prvků na webové stránce pomocí kaskádových stylů. Následuje definice vzhledu grafických zobrazovačů, které jsou nastaveny podle zadané veličiny. Dále jsou definovány jednotlivé blokové elementy a formuláře, pomocí kterých získáváme zadávané hodnoty uživatelem ve vizualizačním prostředí. Následuje ukázka formuláře pro získání požadované hodnoty proudu.

```
<div id="SetCurrent">
        <form name="SetCurrForm">
                <INPUT    type="number"    name="DesiredCurrent"    size="6"
                maxlength="6"
                min="-23.04" max="23.04" step="0.01" value="0.00"/>
        </form>
</div>
```

Poslední část kódu vizualizačního prostředí je psaná JavaScript-em. V této části kódu se řeší všechny přepočty proměnných, odesílání a přijímání hodnot proměnných do mikrokontroleru, blokování tlačítek při vypnutém měniči a obnovování zobrazovačů s periodou 500ms.

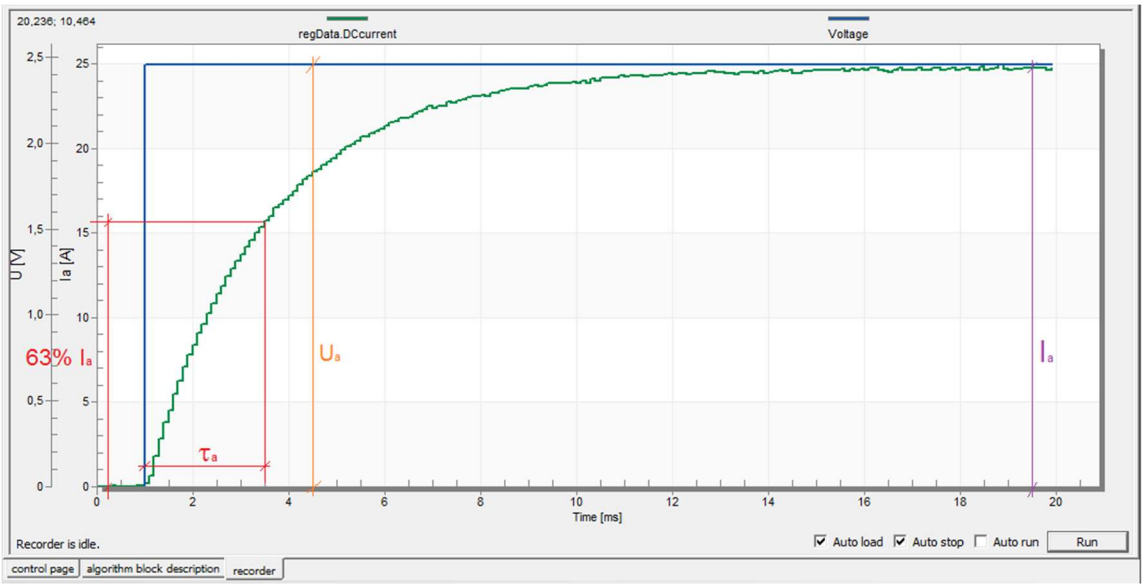
V následující funkci si ukážeme, jakým způsobem získáme požadovanou hodnotu proudu z formuláře po zmáčknutí tlačítka *Set current*.

```
function SetDesCurr_fce() {
        var descurrent;
        descurrent =
        (SetCurrForm.elements["DesiredCurrent"].value)/185;
        if (isNaN(descurrent))
                window.alert("Value is not a number !");
        else if((descurrent > 0.135135136)||(descurrent < -
        0.135135136))
                window.alert("Maximum limit reached ! Max is +-25A");
        else
        fmstr.WriteVariable("regData.DCcurrentDesired",descurrent);
}
```

Funkce *SetDesCurr_fce()* je volána po zmáčknutí tlačítka ve vizualizačním prostředí. Následně se získá proměnná z formuláře, která je dále pomocí směrnice přepočítána na rozsah frakční proměnné. Následuje testování hodnoty proměnní, zda je v zadaném rozsahu ± 25A. V případě, že proměnná je v rozsahu, odešle se do mikrokontroleru.

6.2.3 Návrh regulátoru proudu

Pro návrh regulátoru byla použita metoda optimálního modulu, jelikož je soustava statická, tzn. není integrační. Jedná se o soustavu druhého řádu s jednou velkou časovou konstantou a jednou malou časovou konstantou. Vhodným regulátorem podle metody optimálního modulu je PI regulátor, který kompenzuje velkou časovou konstantu soustavy.



Obrázek 26 Měření elektromagnetické časové konstanty stejnosměrného motoru

Pro změření elektromagnetické konstanty a odporu vinutí musíme zajistit nulovou hodnotu indukovaného napětí v rotoru tak, že rotor zabrzdíme. Po přivedení napětí na mechanicky blokovanou kotvu motoru zjistíme v ustáleném stavu proud protékající odporem R_a. Z hodnoty napětí a proudu vypočteme pomocí Ohmova zákona odpor kotvy. Elektromagnetickou časovou konstantu zjistíme z přechodové charakteristiky (Obrázek 26) tak, že zjistíme čas, ve kterém je hodnota proudu rovna 63 % ustálené hodnoty proudu. Časovou konstantu τ_a vypočteme rozdílem času, kdy je hodnota proudu 63 % ustálené hodnoty proudu a času skoku napětí na kotvě.

Pro přesnější hodnotu elektromagnetické konstanty se musí změřit hodnoty několikrát, jelikož se trochu mění s aktuální polohou natočení rotoru.

Tabulka 4 Naměřené a vypočítané hodnoty napětí, proudu a časové konstanty pro výpočet regulátoru

	U_a	I_a	τ_a	R_a	L_a
	[V]	[A]	[ms]	[mΩ]	[μH]
1	2,460	21,770	2,521	113,000	284,872
2	2,460	23,340	2,884	105,398	303,969
3	2,460	27,208	3,157	90,415	285,439
4	2,460	22,711	2,793	108,318	302,531
5	2,460	24,240	2,323	101,485	235,750
6	2,460	25,589	2,748	96,135	264,179
7	2,460	22,995	2,816	106,980	301,255
8	2,460	24,869	2,680	98,918	265,101
Φ	2,460	24,090	2,740	102,581	280,387

Výpočet odporu kotvy pomoci Ohmovy metody:

$$R_a = \frac{U_a}{I_a} = \frac{2{,}460}{21{,}770} = 113\ m\Omega \tag{6}$$

Výpočet indukčnosti kotvy pomocí elektromagnetické konstanty:

$$L_a = R_a * \tau_a = 113 * 10^{-3} * 2{,}521 * 10^{-3} = 284{,}872\ \mu H \tag{7}$$

6.2.4 Výpočet regulátoru proudu

Zesílení snímače proudu:

$$K_{či} = \frac{výstup}{vstupu} = \frac{\frac{80}{185}}{80} = \frac{1}{185} \equiv 5{,}405 * 10^{-3}\,\frac{1}{A} \tag{8}$$

Zesílení čtyř-kvadrantového měniče:

$$K_{mě} = \frac{\Delta U_D}{\Delta U_{ř}} = \frac{34}{0{,}5} = 68 - \tag{9}$$

Časová konstanta měniče:

$$\tau_{mě} = \frac{1}{2 * f_{PWM}} = \frac{1}{2 * 10000} = 5 * 10^{-5}\ s \tag{10}$$

Zesílení soustavy:

$$K_S = \frac{K_{mě} * K_{či}}{R_a} = \frac{68 * 5{,}405 * 10^{-3}}{102{,}581 * 10^{-3}} = 3{,}583 - \tag{11}$$

Přenos soustavy:

$$\begin{aligned} F_s(p) &= \frac{K_s}{(1 + p\tau_{mě}) * (1 + p\tau_a)} \\ &= \frac{3{,}583}{(1 + p * 5 * 10^{-5}) * (1 + p * 2{,}74 * 10^{-3})} \end{aligned} \tag{12}$$

Přenos regulátoru podle metody OM:

$$\begin{aligned} F_R(p) &= F_o(p) * \frac{1}{F_s(p)} \\ &= \frac{1}{2p\tau_\sigma(1 + p\tau_\sigma)} * \frac{(1 + p\tau_{mě}) * (1 + p\tau_a)}{K_s} \end{aligned} \tag{13}$$

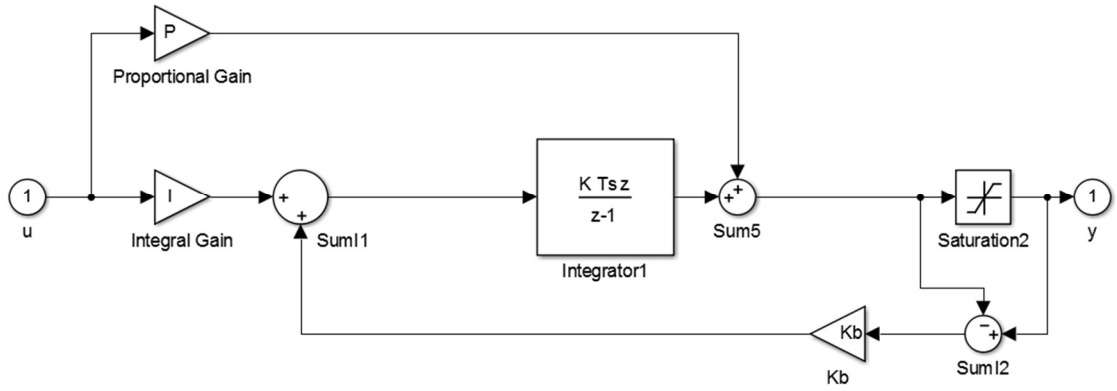
Jelikož časová konstanta měniče $\tau_{mě}$ je nejmenší časová konstanta soustavy, tak ji dosadíme za časovou konstantu τ_σ.

$$\begin{aligned} F_R(p) = F_o(p) * \frac{1}{F_s(p)} &= \frac{1}{2p\tau_{mě}} * \frac{(1 + p\tau_a)}{K_s} \\ &= \frac{1 + p * 2{,}74 * 10^{-3}}{2p * 5 * 10^{-5} * 3{,}583} \\ &= \frac{2790{,}9572}{p} + 7{,}6472 \end{aligned} \tag{14}$$

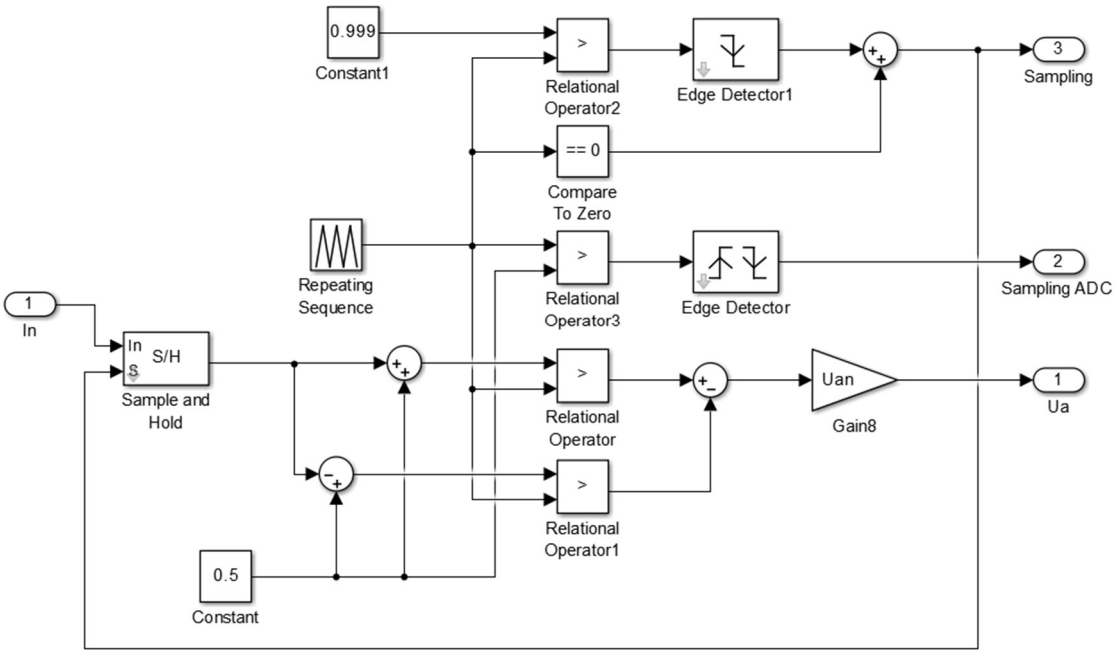
Z předchozího výpočtu vyplývá, že proporcionální zesílení K_p je 7,6472 a integrační zesílení K_I je 2790,9572.

6.2.5 Ověření výpočtu PI regulátoru proudu v programu Simulink

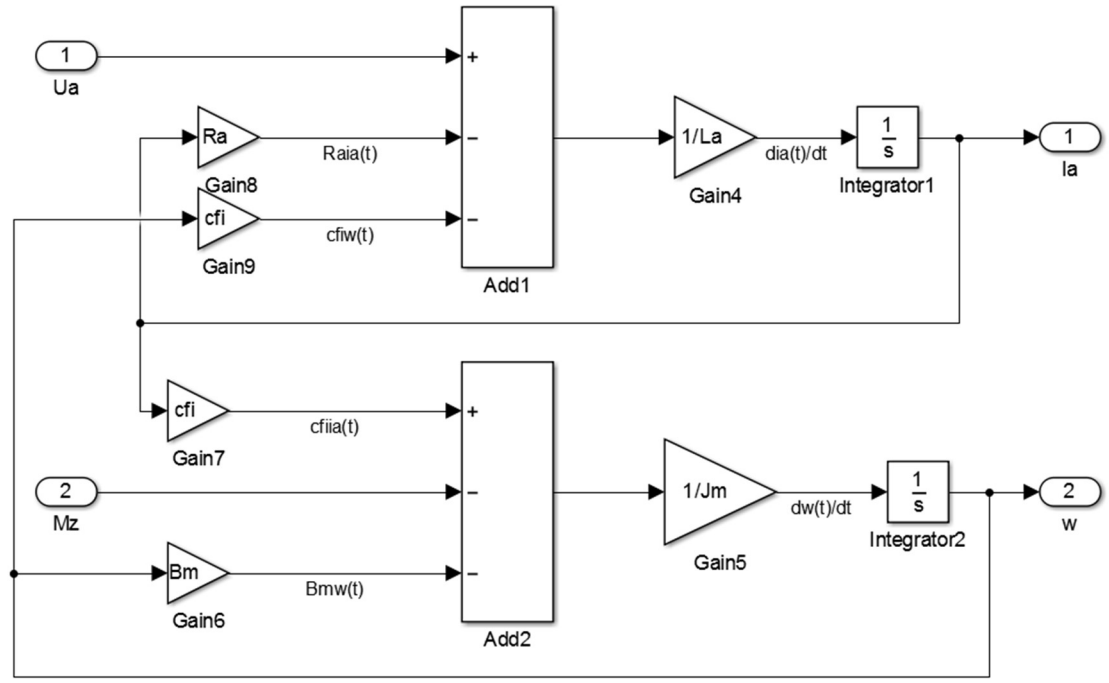
Pro přesnou simulaci musel být sestaven přesný model reálného provedení. V této práci je použito unipolární řízení, protože na motoru je dvojnásobná frekvence proudu, než má proud vytékající z čtyř-kvadrantového měniče. Z důvodu vyšší frekvence proudu v kotvě motoru má zvlnění proudu menší hodnotu.



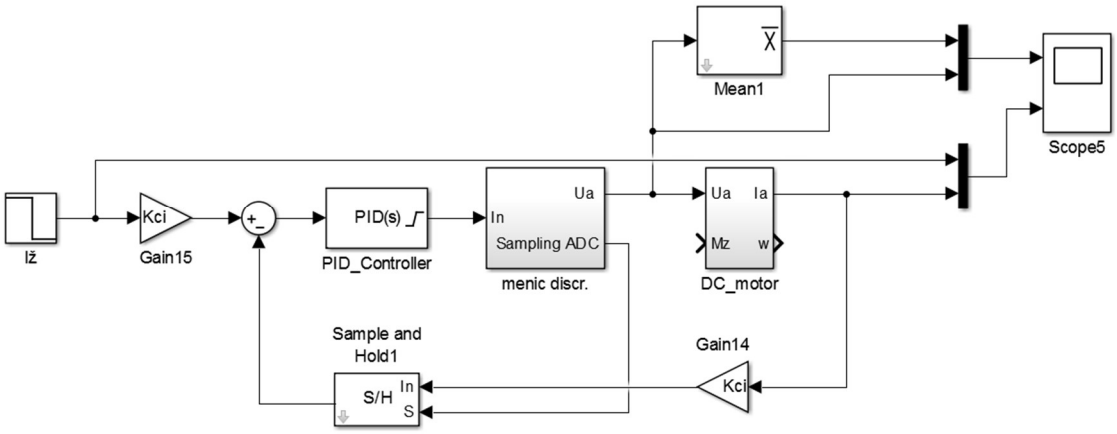
Obrázek 27 Schéma paralelního PI regulátoru



Obrázek 28 Schématické zobrazení funkce číslicového PWM modulátoru s unipolárním řízením.



Obrázek 29 Matematický model stejnosměrného motoru.

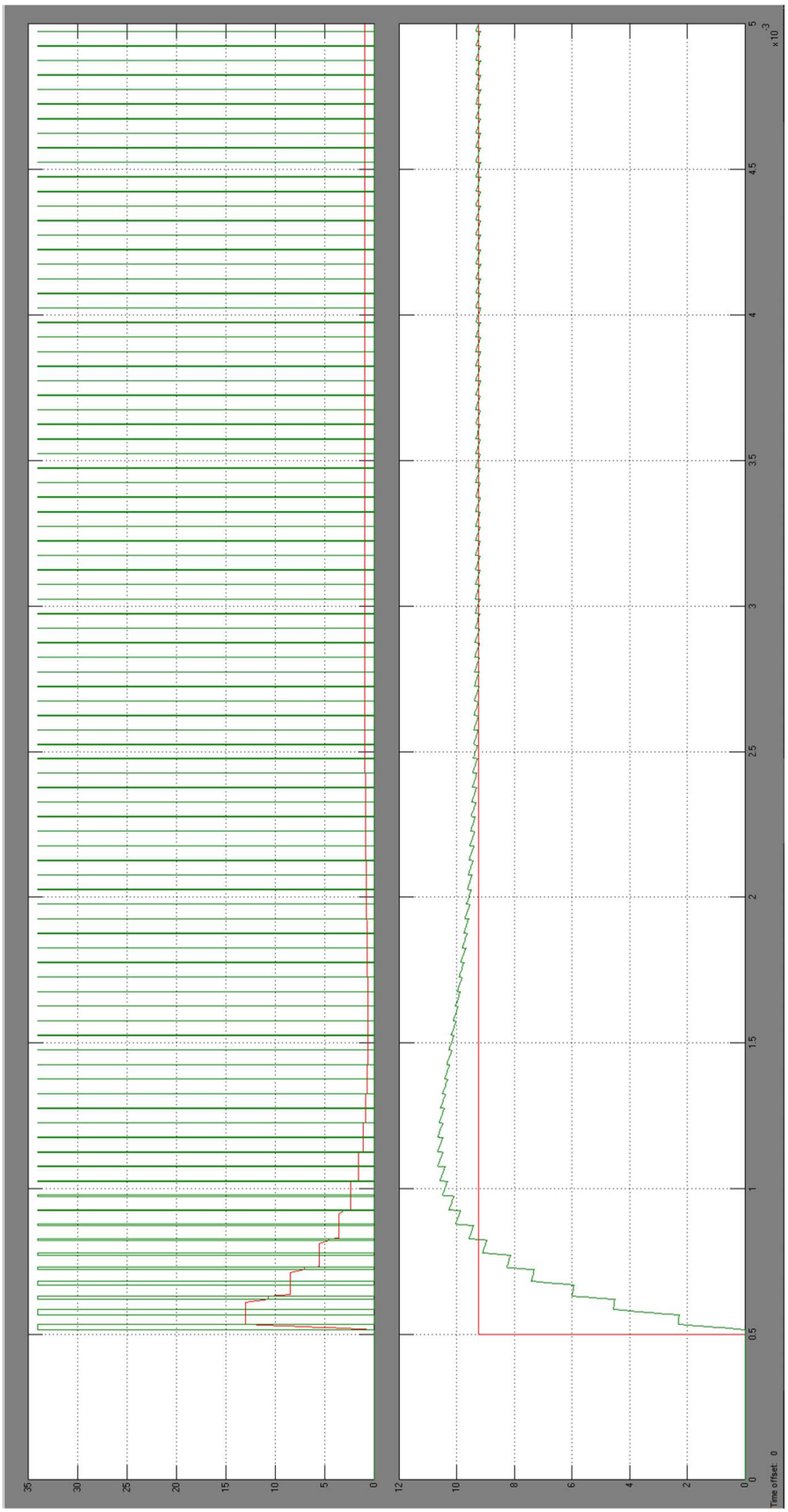


Obrázek 30 Blokové schéma simulace regulátoru proudu stejnosměrného motoru.

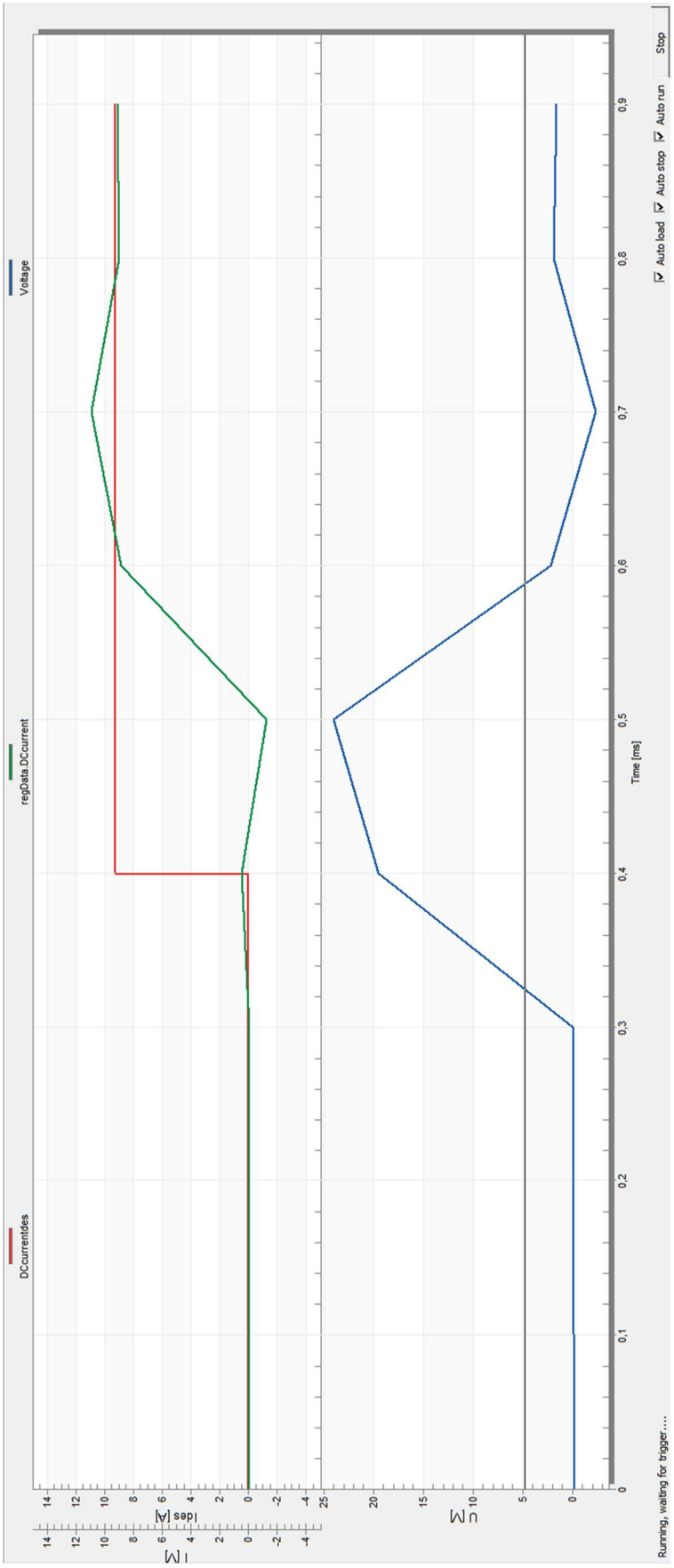
Na následujícím obrázku (Obrázek 31) jsou zobrazeny dva grafy:

- V prvním grafu je vidět průběh střední hodnoty napětí na motoru (červená křivka), dále průběh pulsní šířkově modulovaného napětí U_a (zelená barva).
- V druhém grafu je vidět průběh proudu kotvou (zelená křivka) na žádané hodnotě proudu (červená křivka). Odezva skutečného regulátoru je stejná jako odezva regulátoru při simulaci.

Na obrázku (Obrázek 32) je vidět skutečná odezva regulátoru proudu stejnosměrného motoru na standu.



Obrázek 31 Simulace odezvy regulátoru proudu na skok žádané hodnoty proudu v programu Simulink.



Obrázek 32 Odezva regulátoru proudu na skok žádané hodnoty proudu.

6.2.6 Výpočet momentu stejnosměrného motoru

Pro výpočet momentu stejnosměrného motoru s permanentními magnety musíme znát konstantu stroje $c\phi$. Tuto konstantu lze vyjádřit ze základních rovnic stejnosměrného stroje (15 a 16).

$$U_i = c\phi\omega \tag{15}$$

$$M = c\phi I_a \tag{16}$$

Nyní si první rovnici upravíme:

$$c\phi = \frac{U_i}{\omega} = \frac{U_i \cdot 60}{2\pi n} \tag{17}$$

Z předchozího vztahu je vidět, že pro výpočet konstanty potřebujeme znát indukované napětí U_i a rychlost otáčení rotoru n. Nyní můžeme provést měření. Soustrojí je potřeba roztočit, proto si zvolíme libovolné otáčky indukčního motoru při zablokovaném výstupu měniče pro stejnosměrný motor. Následně je možné pomocí multimetru změřit napětí na svorkách stejnosměrného motoru. Toto napětí je přibližně rovno indukovanému napětí. Z hodnot otáček a indukovaného napětí spočítáme pomocí rovnice (17) konstantu stroje. Nyní je možné vypočítat moment stejnosměrného stroje pomocí rovnice (16).

7 ZÁVĚR

Úkolem této práce bylo navrhnout firmware pro dva mikrokontrolery, vizualizační prostředí, se kterým budou studenti následně pracovat. V této práci byl také vypracovaný návod pro použití v laboratořích. Dalším úkolem bylo popsat hardware laboratorního standu, navrhnout strukturu softwaru s ohledem na čtyři laboratorní úlohy, přičemž hlavní podmínkou bylo to, aby se nemusely pro každou úlohu přeprogramovávat mikroprocesory pohonů (měniče).

V této práci je popsaná hardwarová struktura výukového laboratorního standu, dále bylo vytvořeno blokové schéma standu z poskytnutých schémat standu. Dále je popsána komunikace mezi oběma mikrokontrolery. V následující části práce jsou popsány způsoby řízení indukčního motoru včetně Clarkové a Parkovy transformace a modulace prostorového vektoru napětí. Byl vytvořen a popsán firmware pro oba pohony, naprogramované vizualizační prostředí pro řízení měničů, nastavení jednotlivých parametrů, měření jednotlivých veličin a zobrazování průběhů atd. V této práci je vytvořena první laboratorní úloha, ve které se studenti naučí navrhovat a nastavovat PI regulátor proudu.

Při programování firmwaru pro měniče bylo zjištěno několik problémů s hardwarem. Prvním problém byl ten, že nefungovala komunikace s R/D převodníkem. Jelikož část kódu pro komunikace s R/D převodníkem byla správná, hledal se problém na straně hardwaru pomocí osciloskopu. Měřením bylo zjištěno, že je přerušená cesta mezi procesorem a R/D převodníkem. Dalším problémem bylo to, že se měnič během zkoušek samovolně vypínal. To bylo způsobeno vybavováním saturační ochrany jednoho z výkonových tranzistorů. Při měření této závady bylo zjištěno, že je špatně nastavená saturační ochrana, proto je ve měničích tato ochrana prozatím vypnuta. Dalším problémem byla nedostatečná kvalita přenesených zpráv mezi mikrokontrolery. Tento problém byl vyřešen změnou komunikační periférie z SCI na MSCAN.

Následně je zpracován výpočet PI regulátoru proudu, který byl ověřen na standu. Jelikož byl při měření zjištěn větší překmit, než má mít metoda optimálního modulu, byl vytvořen model měniče a diskrétního PI regulátoru včetně vzorkování signálu ze snímače proudu v programu Simulink, který respektuje co nejvíce detailů. V této simulaci bylo zjištěno, že měřený překmit proudu nad žádanou hodnotu je totožný s překmitem proudu u měniče stejnosměrného motoru.

Pro účel prezentace na veletrhu AMPER 2017 byl navrhnut demoprogram, který sloužil k demonstraci funkce standu a byl spuštěn v nekonečné smyčce. V tomto demoprogramu byl předveden regulátor otáček stejnosměrného motoru a zatěžování asynchronního motoru v režimu *U/f* stejnosměrným motorem.

Na přiloženém CD jsou všechny potřebné zdrojové kódy pro firmware obou pohonů a všechny potřebné projekty.

Literatura

[1] HUDÁK, Ondřej. *Laboratorního soustrojí s asynchronním a stejnosměrným motorem*. Brno, 2012. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Dalibor Červinka, Ph.D.

[2] ONDREJČEK, Vladimír. *Řídicí systém laboratorního standu pro výukové účely*. Brno, 2014. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Dalibor Červinka, Ph.D.

[3] DRÁB, Domink. *Číslicově řízený měnič pro asynchronní motor*. Brno, 2015. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav výkonové elektrotechniky a elektroniky. Vedoucí práce Ing. Jan Knobloch.

[4] ANALOG DEVICES, . *AD2S1200 datasheet: 12-Bit R/D Converter with Reference Oscillator* [online]. 2003 [cit. 2016-12-23]. Dostupné z: http://www.analog.com/media/en/technical-documentation/data-sheets/AD2S1200.pdf

[5] AVAGO TECHNOLOGIES, . *ACPL-333J datasheet: 2,5 Amp Output Current IGBT Gate Driver Optocoupler with Integrated Desaturation Detection* [online]. 2015 [cit. 2016-12-23]. Dostupné z: https://docs.broadcom.com/docs/AV02-1087EN

[6] FREESCALE SEMICONDUCTOR, . *MC56F827xx datasheet: Reference Manual* [online]. 2013 [cit. 2016-12-23]. Dostupné z: http://cache.freescale.com/files/32bit/doc/ref_manual/MC56F827XXRM.pdf

[7] NXP, . *PCA82C251: CAN transceiver for 24V systems* [online]. 25. srpna 2011. NXP, 2011 [cit. 2017-05-07]. Dostupné z: http://www.nxp.com/documents/data_sheet/PCA82C251.pdf

[8] BREJL, Milan, Michal PRINC a Petr UHLIR. *AC Induction Motor Volts per Hertz Control with Speed Closed Loop: Driven by eTPU on MPC550* [online]. 2006 [cit. 2016-12-27]. Dostupné z: http://www.nxp.com/assets/documents/data/en/application-notes/AN3205.pdf

[9] RYDLO, Pavel. *Řízení elektrických střídavých pohonů*. Vyd. 2. V Liberci: Technická univerzita, 2007. ISBN 9788073722234.

[10] KLÍMA, Bohumil. *Mikroprocesorové řízení elektrických pohonů*. FEKT VUT v Brně, 2014.

[11] FREESCALE SEMICONDUCTOR, . *Advanced Control Library: User Reference Manual* [online]. 2014 [cit. 2016-12-24]. Dostupné z: http://www.nxp.com/assets/documents/data/en/user-guides/56800Ex_ACLIB.pdf

[12] FREESCALE SEMICONDUCTOR, . *Math Library: User Reference Manual* [online]. 2014 [cit. 2016-12-24]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/user_guide/56800Ex_MLIB.pdf

[13] FREESCALE SEMICONDUCTOR, . *Motor Control Library: User Reference Manual* [online]. 2014 [cit. 2016-12-24]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/user_guide/56800Ex_MCLIB.pdf

[14] FREESCALE SEMICONDUCTOR, . *General Functions Library: User Reference Manual* [online]. 2014 [cit. 2016-12-24]. Dostupné z: http://www.nxp.com/assets/documents/data/en/user-guides/56800Ex_GFLIB.pdf

[15] FREESCALE SEMICONDUCTOR, . *MC56F827xx: Reference Manual* [online]. 2013 [cit. 2016-12-25]. Dostupné z: http://cache.nxp.com/assets/documents/data/en/reference-manuals/MC56F827XXRM.pdf

[16] HANÁK, Michal. *FreeMaster for Embedded Applications User Manual: Real_Time Monitor, Control Panel and Demostration Tool* [online]. 2004 [cit. 2016-12-26]. Dostupné z: http://www.nxp.com/assets/documents/data/en/user-guides/FreeMasterUG.pdf

[17] FREESCALE INC., . *FreeMASTER for Embedded Applications: User Guide* [online]. Rev. 2.4 06/2014. NXP, 2014 [cit. 2017-05-14]. Dostupné z: http://www.nxp.com/assets/documents/data/en/user-guides/FMSTERUG.pdf

[18] HTML Canvas Gauges v2.1. *Github, Inc* [online]. 2017 [cit. 2017-05-21]. Dostupné z: https://github.com/Mikhus/canvas-gauges

Seznam symbolů, veličin a zkratek

$2p$	[-]	počet pólů
P_n	[W]	jmenovitý výkon
U_n	[V]	jmenovité napětí
I_n	[A]	jmenovitý proud
n_n	[min^{-1}]	jmenovité otáčky
n_0	[min^{-1}]	otáčky na prázdno
N_n	[Nm]	jmenovitý moment
η	[%]	účinnost
$cos\varphi$	[-]	účiník
M_z/M_n	[-]	poměrný záběrný proud
I_k/I_n	[-]	poměrný záběrný proud
M_{max}/M_n	[-]	poměrný moment zvratu
τ_a	[s]	elektromagnetická časová konstanta
τ_m	[s]	elektromechanická časová konstanta
τ_σ	[s]	malá časová konstanta soustavy
U_a	[V]	napětí kotvy stejnosměrného motoru
I_a	[A]	proud kotvou stejnosměrného motoru
R_a	[Ω]	odpor kotvy stejnosměrného motoru
L_a	[H]	indukčnost kotvy stejnosměrného motoru
$K_{či}$	[A^{-1}]	zesílení čidla proudu
$K_{mě}$	[-]	zesílení měniče
K_s	[-]	zesílení soustavy
K_I	[-]	integrační zesílení regulátoru
K_p	[-]	proporcionální zesílení regulátoru
T	[s]	perioda
$c\phi$	[Wb/rad]	konstanta stejnosměrného stroje
U_i	[V]	indukované napětí kotvy
ω	[rad/s]	úhlová rychlost
ω_s	[rad/s]	synchronní úhlová rychlost
$\boldsymbol{i}$	[A]	komplexní prostorový vektor proudu
$\boldsymbol{u}$	[V]	komplexní prostorový vektor napětí
$\boldsymbol{\psi}$	[Wb]	komplexní prostorový vektor magnetického toku
υ_K	[rad]	transformační úhel souřadných systémů
f_{PWM}	[Hz]	frekvence PWM
f_{stat}	[-]	frekvence statorového pole
$f_{otHř}$	[-]	frekvence otáčení hřídele
s	[-]	skluz indukčního motoru

Indexy

R	prostorový vektor, složka prostorového vektoru rotorové veličiny
s	prostorový vektor, složka prostorového vektoru statorové veličiny
xy	prostorový vektor v obecném souřadném systému
α	reálná složka prostorového vektoru ve statorovém souřadném systému
β	imaginární složka prostorového vektoru ve statorovém souřadném systému
$\alpha\beta$	prostorový vektor ve statorovém souřadném systému

Zkratky

F_o	přenos otevřené smyčky
F_R	přenos regulátoru
F_s	přenos soustavy
R/D	resolver / digital převodník
SPI	sériové periferní rozhraní (Seriál Peripheral Interface)
SCI	sériové komunikační rozhraní (Seriál Communications Interface)
DC/DC	stejnosměrný měnič
PWM	pulsní šířková modulace (Pulse Width Modulation)
OM	optimální modul
SVM	modulace prostorového vektoru (Space Vector Modulation)
FSLESL	knihovny pro řízení motorů (Freescale Embedded Software Libraries)
MCU	mikroprocesor / mikrokontroler (Miceocontroller Unit)
I/O	vstupně / výstupní porty (Input / Output)
PID	proporcionálně Integračně Derivační regulátor
PI	proporcionálně Integrační regulátor
A/D, ADC	analogově digitální převodník (Analog to Digital converter)
MOS-FET	unipolární tranzistor (Metal Oxide Semiconductor Field Effect Transistor)
DMA	přímý přístup k paměti (Direct Memory Access)
COM port	komunikační port (Communication port)
CSS	kaskádové styly (Cascading Style Sheets)
DSP	digitální signálový procesor (digital signal processor)
CAN	vnitřní komunikační síť senzorů a funkčních jednotek (controller area network)
MSCAN	Modular / Scalable Controller Area Network
ID	identifikátor

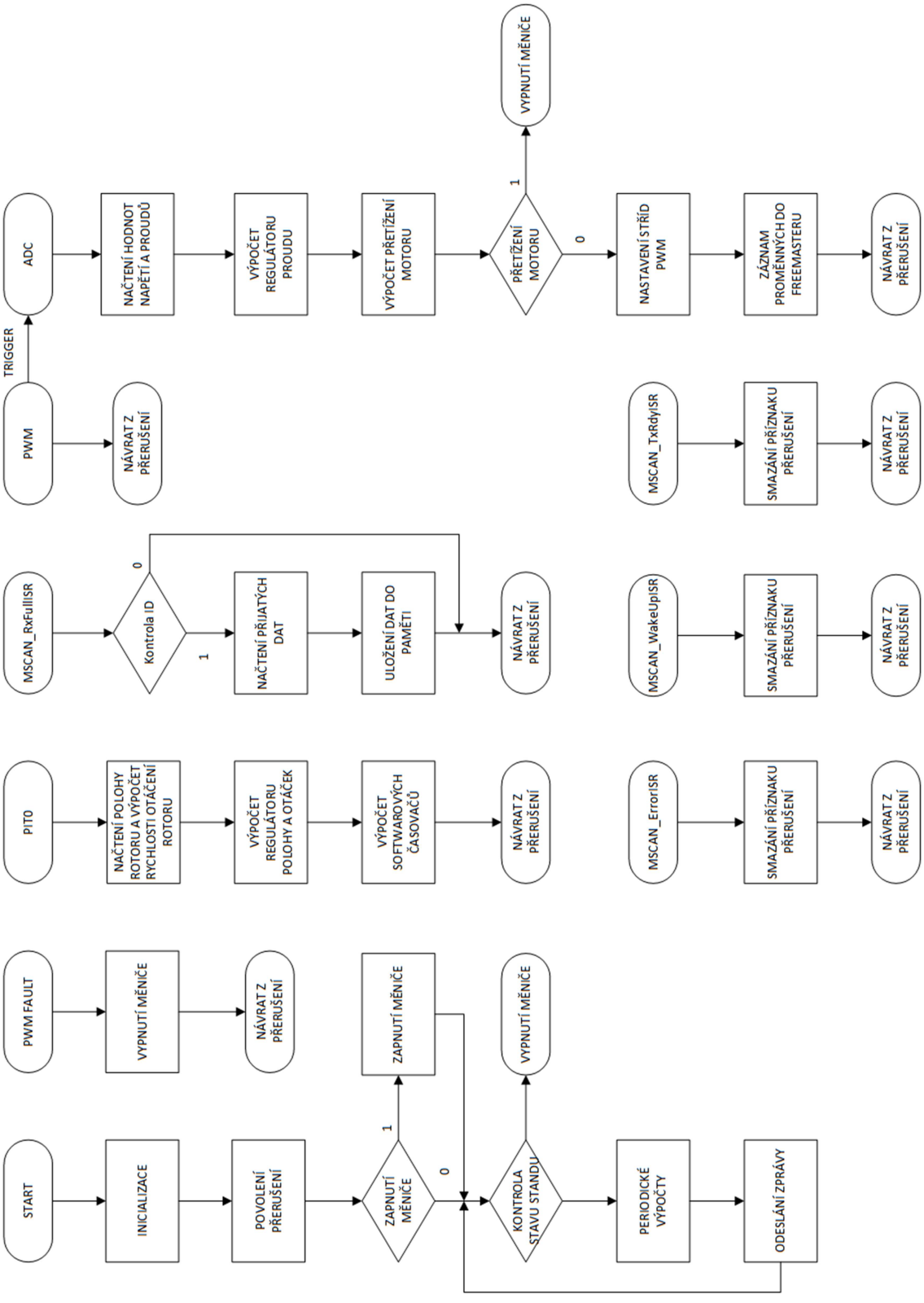
Seznam příloh

A. Definice vstupně/výstupních portů

```
#ifndef PROCESSOR_PINS_H_
#define PROCESSOR_PINS_H_

// PWM
#define GPIO_PWM              GPIO_E
#define PWM0A                 BIT_1       //PWM horní tranzistor větve A
#define PWM0B                 BIT_0       //PWM dolní tranzistor větve A
#define PWM1A                 BIT_3       //PWM horní tranzistor větve B
#define PWM1B                 BIT_2       //PWM dolní tranzistor větve B
#define PWM_FAULT             BIT_6       //Signál z fault budičů

//I2C
#define GPIO_I2C              GPIO_F
#define SDA                   BIT_3       //I2C SDA pro LCD displej
#define SCL                   BIT_2       //I2C SCL pro LCD displej

//Serial
#define GPIO_SERIAL           GPIO_C
#define RXD                   BIT_12      //Sériová komunikace RXD
#define TXD                   BIT_11      //Sériová komunikace TXD

//resolver
#define GPIO_POS_SPEED        GPIO_F
#define POS_SPEED             BIT0        //Resolver - Poloha / rychlost

#define GPIO_RESOLVER         GPIO_C
#define RES_SAMPLE            BIT_10      //Resolver - SAMPLE
#define RES_SCLK              BIT_9       //Resolver - serial clock
input
#define RES_SERIAL_OUT        BIT_8       //Resolver - serial output
#define RES_READ              BIT_7       //Resolver - READ

//Blocking drivers
#define GPIO_EN_DRIVERS       GPIO_C
#define EN_DRIVERS            BIT_6       //Blokování budičů - ENABLE

//Encoder
#define GPIO_ENC_MV           GPIO_A
#define ENC_MV                BIT_7       //Encoder Index

#define GPIO_ENC_AB           GPIO_C
#define ENC_A                 BIT_3       //Encoder A
#define ENC_B                 BIT_4       //Encoder B

//Analog input
#define GPIO_ANALOG           GPIO_A
#define ANA_U1                BIT_6       //Napětí meziobvodu
#define ANA_I2                BIT_5       //Proud měničem větev A
#define ANA_I1                BIT_4       //Proud měničem větev C

//Procesor LED
#define GPIO_LED              GPIO_C
#define LED                   BIT_13      //Signalizační LED
```

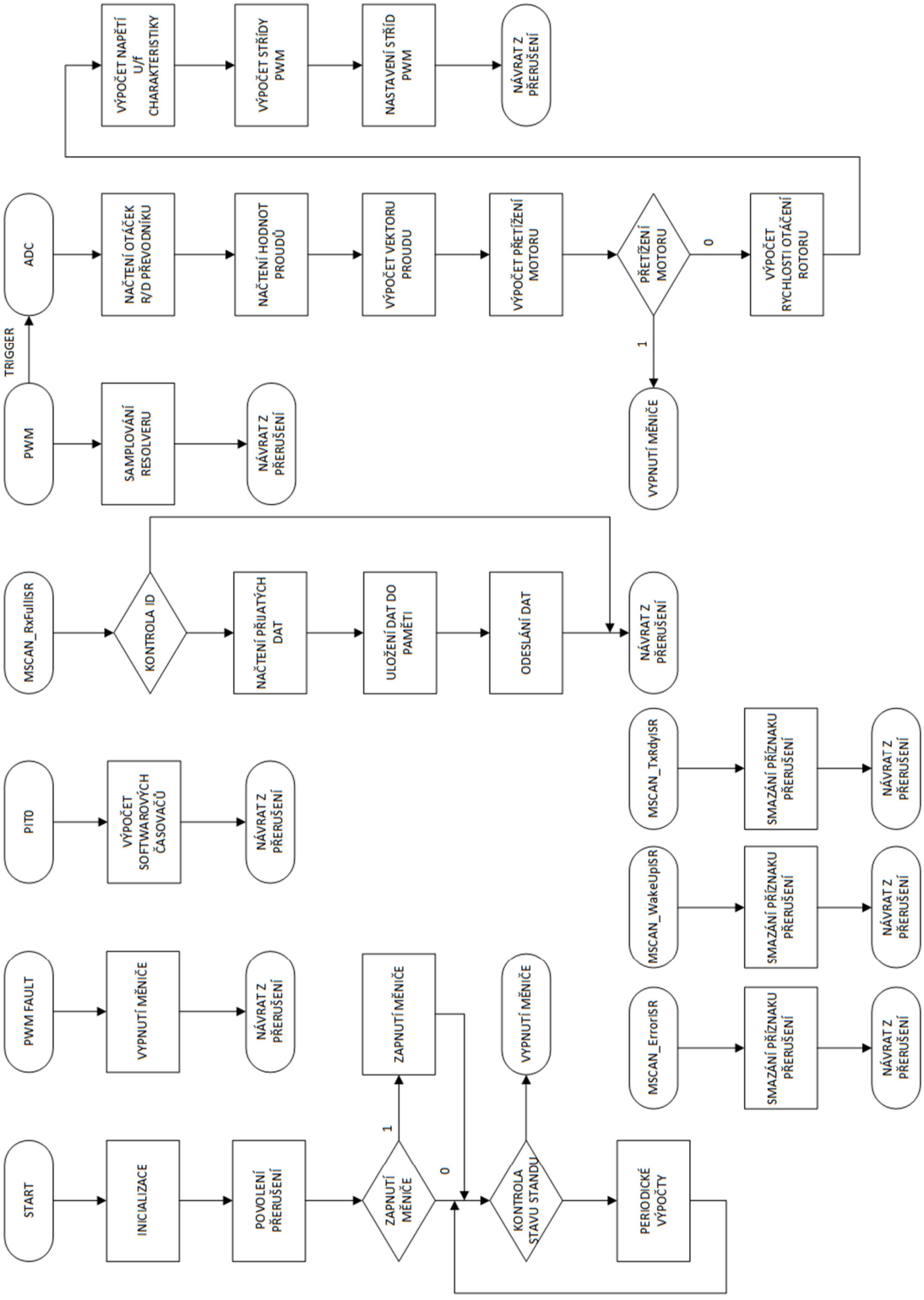
```
#define GPIO_PWMLED           GPIO_E
#define LED6                  BIT_7
#define LED5                  BIT_4
#define LED4                  BIT_5

//Switch
#define GPIO_SW12             GPIO_C
#define SWITCH_1              BIT_15            //tlačítko 1
#define SWITCH_2              BIT_14            //tlačítko 2
#define GPIO_SW3              GPIO_F
#define SWITCH_3              BIT_7             //tlačítko 3

#define PWM_3F_ALL      EFPWM_SUB0_PWM_A |EFPWM_SUB0_PWM_B|
      EFPWM_SUB1_PWM_A |EFPWM_SUB1_PWM_B |
      EFPWM_SUB2_PWM_A|EFPWM_SUB2_PWM_B
#define PWM_SS_ALL      EFPWM_SUB0_PWM_A
      |EFPWM_SUB0_PWM_B|EFPWM_SUB1_PWM_A| EFPWM_SUB1_PWM_B
#endif /* PROCESSOR_PINS_H_ */
```

B. Zjednodušená struktura firmware pro stejnosměrný motor



Obrázek 33 Zjednodušená struktura firmware pro stejnosměrný motor

C. Zjednodušená struktura firmware pro stejnosměrný motor



Obrázek 34 Zjednodušená struktura firmware pro stejnosměrný motor

D. Obsah přiloženého CD

Přiložené CD obsahuje následující soubory:

- Projekt s firmwarem pro stejnosměrný motor
- Projekt s firmwarem pro indukční motor
- Projekt se zdrojovým kódem vizualizace pro program FreeMASTER
- Laboratorní návod první úlohy
- Projekt s návrhem převodníku CAN